

# Práctica Hive + Impala + HDFS + Spark

- A partir de los datos (CSV) de Padrón de Madrid (<https://datos.madrid.es/egob/catalogo/200076-1-padron.csv>) llevar a cabo lo siguiente:

## 1- Creación de tablas en formato texto.

- 1.1)

Crear Base de datos "datos\_padron".

```
hive> create database datos_padron
> ;
OK
```

- 1.2)

Crear la tabla de datos padron\_txt con todos los campos del fichero CSV y cargar los datos mediante el comando LOAD DATA LOCAL

INPATH. La tabla tendrá formato texto y tendrá como delimitador de campo el caracter ';' y los campos que en el documento original están encerrados en comillas dobles "" no deben estar envueltos en estos caracteres en la tabla de Hive (es importante indicar esto utilizando el serde de OpenCSV, si no la importación de las variables que hemos indicado como numéricas fracasará ya que al estar envueltos en comillas los toma como strings) y se deberá omitir la cabecera del fichero de datos al crear la tabla.

```
hive> create table datos_padron_txt (
  > COD_DISTrito int,
  > DESC_DISTrito string,
  > COD_DIST_BARRIO int,
  > DESC_BARRIO string,
  > COD_BARRIO int,
  > COD_DIST_SECCION int,
  > COD_SECCION int,
  > COD_EDAD_INT int,
  > EspanolesHombres int,
  > EspanolesMujeres int,
  > ExtranjerosHombres int,
  > ExtranjerosMujeres int)
  > ROW FORMAT SERDE
  > 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  > WITH SERDEPROPERTIES (
  > "separatorChar" = "\u0059",
  > "quoteChar" = "\"",
  > "escapeChar" = "\\"
  > )
  > STORED AS TEXTFILE;
OK
Time taken: 0.307 seconds
```

```
hive> load data inpath '/user/cloudera/padron/Rango_Edades_Seccion_202104.csv' into table datos_padron_txt;
Loading data to table default.datos_padron_txt
Table default.datos_padron_txt stats: [numFiles=1, totalSize=22586014]
OK
Time taken: 2.454 seconds
```

- 1.3)

Hacer trim sobre los datos para eliminar los espacios innecesarios guardando la tabla resultado como padron\_txt\_2. (Este apartado se puede hacer creando la tabla con una sentencia CTAS.)

```
hive> create table padron_txt_2 stored as textfile
> tblproperties("skip.header.line.count" = "1") as
> select cod_distrito, trim(desc_distrito) desc_distrito,
> cod_dist_barrio, trim(desc_barrio) desc_barrio,
> cod_barrio, cod_dist_seccion, cod_seccion, cod_edad_int,
> case when length(espanoleshombres) = 0 then '0' else espanoleshombres end as espanoleshombres,
> case when length(espanolesmujeres) = 0 then '0' else espanolesmujeres end as espanolesmujeres,
> case when length(extranjeroshombres) = 0 then '0' else extranjeroshombres end as extranjeroshombres,
> case when length(extranjerosmujeres) = 0 then '0' else extranjerosmujeres end as extranjerosmujeres
> from padron_txt;
Query ID = cloudera_20210421134848_c917c3e2-1aac-4610-97b5-0abac0e51c9d
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1617092985514_0030, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1617092985514_0030
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1617092985514_0030
```

```
hive> select * from padron_txt_2 limit 5;
OK
COD_DISTrito    DESC_DISTrito    COD_DIST_BARRIO  DESC_BARRIO      COD_BARRIO      COD_SECCION      COD_EDAD_INT     EspanolesHombres  EspanolesMujeres  extranjerosMujeres
1      CENTRO    101      PALACIO 1      1006      6      97      1
1      CENTRO    101      PALACIO 1      1006      6      98      1      1
1      CENTRO    101      PALACIO 1      1006      6      99      1
1      CENTRO    101      PALACIO 1      1006      6      100     1
Time taken: 0.196 seconds, Fetched: 5 row(s)
hive>
```

- 1.4)

Investigar y entender la diferencia de incluir la palabra LOCAL en el comando LOAD DATA.

La palabra “Local” se usa para cuando el fichero se encuentra en tu sistema de archivos local y no en HDFS, en mi caso no la he usado porque antes he copiado el csv a HDFS.

- 1.5)

En este momento te habrás dado cuenta de un aspecto importante, los datos nulos de nuestras tablas vienen representados por un espacio vacío y no por un identificador de nulos comprensible para la tabla. Esto puede ser un problema para el tratamiento posterior de los datos. Podrías solucionar esto creando una nueva tabla utilizando sentencias case when que sustituyan espacios en blanco por 0. Para esto primero comprobaremos que solo hay espacios en blanco en las variables numéricas correspondientes a las últimas 4 variables de nuestra tabla (podemos hacerlo con alguna sentencia de HiveQL) y luego aplicaremos las sentencias case when para sustituir por 0 los espacios en blanco. (Pista: es útil darse cuenta de que un espacio vacío es un campo con longitud 0). Haz esto solo para la tabla padron\_txt.

```
hive> select `_c0` from padron_txt_2 where length(`_c0`)=0 limit 5;
OK
Time taken: 0.791 seconds
hive> select * from padron_txt_2 where length(`_c1`)=0 limit 5;
OK
Time taken: 0.224 seconds
hive> select * from padron_txt_2 where length(`_c2`)=0 limit 5;
OK
Time taken: -21.275 seconds
hive> select * from padron_txt_2 where length(`_c3`)=0 limit 5;
OK
Time taken: 0.246 seconds
hive> select * from padron_txt_2 where length(`_c4`)=0 limit 5;
OK
Time taken: 0.295 seconds
hive> select * from padron_txt_2 where length(`_c5`)=0 limit 5;
OK
Time taken: 0.25 seconds
hive> select * from padron_txt_2 where length(`_c6`)=0 limit 5;
OK
Time taken: 0.187 seconds
hive> select * from padron_txt_2 where length(`_c7`)=0 limit 5;
OK
Time taken: 0.24 seconds
```

```
hive> create table padron_txt as
> select `_c0`, `_c1`, `_c2`, `_c3`, `_c4`, `_c5`, `_c6`, `_c7`,
> case when length(`_c8`) = 0 then '0' else `_c8` end as `_c8`,
> case when length(`_c9`) = 0 then '0' else `_c9` end as `_c9`,
> case when length(`_c10`) = 0 then '0' else `_c10` end as `_c10`,
> case when length(`_c11`) = 0 then '0' else `_c11` end as `_c11`
> from padron_txt_2 limit 10;
Query ID = cloudera_20210420141616_03f5870b-40a5-451f-ad2f-8bd310557164
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
```

```
hive> select * from padron_txt limit 5;
OK
1      CENTRO  101      PALACIO 1      1007      7      3      4      1      0      0
1      CENTRO  101      PALACIO 1      1007      7      2      1      3      0      0
1      CENTRO  101      PALACIO 1      1007      7      1      2      2      0      1
1      CENTRO  101      PALACIO 1      1007      7      0      3      1      0      2
1      CENTRO  101      PALACIO 1      1006      6      103     0      1      0      0
Time taken: 0.183 seconds, Fetched: 5 row(s)
```

• 1.6)

Una manera tremendamente potente de solucionar todos los problemas previos (tanto las comillas como los campos vacíos que no son catalogados como null y los espacios innecesarios) es utilizar expresiones regulares (regex) que nos proporciona OpenCSV.

Para ello utilizamos :

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
```

```
WITH SERDEPROPERTIES ('input.regex'='XXXXXXX')
```

Donde XXXXXX representa una expresión regular que debes completar y que identifique el formato exacto con el que debemos interpretar cada una de las filas de nuestro CSV de entrada. Para ello puede ser útil el portal "regex101". Utiliza este método para crear de nuevo la tabla padron\_txt\_2.

Una vez finalizados todos estos apartados deberíamos tener una tabla padron\_txt que conserve los espacios innecesarios, no tenga comillas envolviendo los campos y los campos nulos sean tratados como valor 0 y otra tabla padron\_txt\_2 sin espacios innecesarios, sin comillas envolviendo los campos y con los campos nulos como valor 0. Idealmente esta tabla ha sido creada con las regex de OpenCSV.

```
hive> create table padron_txt (  
  > cod_distrito int,  
  > desc_distrito string,  
  > cod_dist_barrio int,  
  > desc_barrio string,  
  > cod_barrio int,  
  > cod_dist_seccion int,  
  > cod_seccion int,  
  > cod_edad_int int,  
  > espanoleshombres int,  
  > espanolesmujeres int,  
  > extranjeroshombres int,  
  > extranjerosmujeres int)  
  > row format serde  
  > 'org.apache.hadoop.hive.serde2.RegexSerDe'  
  > with serdeproperties ("input.regex" =  
  > '"(\\d*)"\\;"(.*)\\s*"\\;"(\\d*)"\\;"(.*)\\s*"\\;"(\\d*)"\\;"(\\d*)"\\;"  
  > "(\\d*)"\\;"(\\d*)"\\;"(\\d*)"\\;"(\\d*)"\\;"(\\d*)"\\;"(\\d*)"')  
  > stored as textfile  
  > tblproperties("skip.header.line.count" = "1");  
OK  
Time taken: 0.231 seconds
```

```
hive> load data local inpath '/home/cloudera/padron/Rango_Edades_Seccion_202104.csv'  
  > overwrite into table 'datos_padron'. 'padron_txt';  
Loading data to table datos_padron.padron_txt  
Table datos_padron.padron_txt stats: [numFiles=1, numRows=0, totalSize=22586014, rawDataSize=0]  
OK  
Time taken: 8.647 seconds  
hive> select * from padron_txt limit 5;  
OK  
COD_DISTrito  DESC_DISTrito  COD_DIST_BARRIO  DESC_BARRIO  COD_BARRIO  COD_DIST_SECCION  COD_SECCION  
1  CENTRO  101  PALACIO  1  1006  6  97  1  
1  CENTRO  101  PALACIO  1  1006  6  98  1  
1  CENTRO  101  PALACIO  1  1006  6  99  1  
1  CENTRO  101  PALACIO  1  1006  6  100  1  
Time taken: 3.912 seconds, Fetched: 5 row(s)
```

## 2- Investigamos el formato columnar parquet.

- 2.1)

¿Qué es CTAS?

Es una instrucción "Create table as", que permite crear una tabla a partir de otra.

- 2.2)

Crear tabla Hive padron\_parquet (cuyos datos serán almacenados en el formato columnar parquet) a través de la tabla padron\_txt mediante un CTAS.

```
hive> create table padron_parquet stored as parquet
> as select * from padron_txt;
Query ID = cloudera_20210420144242_0727dd72-2796-4589-b531-1261190de372
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1617092985514_0020, Tracking URL = http://quickstart.cloudera
1617092985514_0020/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1617092985514_0020
```

- 2.3)

Crear tabla Hive padron\_parquet\_2 a través de la tabla padron\_txt\_2 mediante un CTAS. En este punto deberíamos tener 4 tablas, 2 en txt (padron\_txt y padron\_txt\_2, la primera con espacios innecesarios y la segunda sin espacios innecesarios) y otras dos tablas en formato parquet (padron\_parquet y padron\_parquet\_2, la primera con espacios y la segunda sin ellos).

```
hive> create table padron_parquet_2 stored as Parquet
> tblproperties("skip.header.line.count" = "1") as
> select cod_distrito, trim(desc_distrito) desc_distrito,
> cod_dist_barrio, trim(desc_barrio) desc_barrio,
> cod_barrio, cod_dist_seccion, cod_seccion, cod_edad_int,
> case when length(espnoleshombres) = 0 then '0' else espnoleshombres end as espnoleshombres,
> case when length(espnolesmujeres) = 0 then '0' else espnolesmujeres end as espnolesmujeres,
> case when length(extranjeroshombres) = 0 then '0' else extranjeroshombres end as extranjeroshombres,
> case when length(extranjerosmujeres) = 0 then '0' else extranjerosmujeres end as extranjerosmujeres
> from padron_parquet;
Query ID = cloudera_20210421134343_3ef2a567-f3a0-4ea6-9bcc-be33920d18f2
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
```

```
hive> select * from default.padron_parquet_2 limit 5;
OK
1      CENTRO  101      PALACIO 1      1006    6      97      1      0      0      0
1      CENTRO  101      PALACIO 1      1006    6      98      1      1      0      0
1      CENTRO  101      PALACIO 1      1006    6      99      0      1      0      0
1      CENTRO  101      PALACIO 1      1006    6      100     0      1      0      0
1      CENTRO  101      PALACIO 1      1006    6      103     0      1      0      0
Time taken: 0.207 seconds, Fetched: 5 row(s)
```

- 2.4)

Opcionalmente también se pueden crear las tablas directamente desde 0 (en lugar de mediante CTAS) en formato parquet igual que lo hicimos para el formato txt incluyendo la sentencia STORED AS PARQUET. Es importante para comparaciones posteriores que la tabla padron\_parquet conserve los espacios innecesarios y la tabla padron\_parquet\_2 no los tenga. Dejo a tu elección cómo hacerlo.

- 2.5)

Investigar en qué consiste el formato columnar parquet y las ventajas de trabajar con este tipo de formatos.

Parquet es un sistema de almacenamiento basado en columnas, que muestra su superioridad cuando nuestra tabla tiene un gran número de columnas y solo vamos a trabajar con un pequeño grupo de ellas. También es capaz de leer en paralelo y la compresión funciona mejor que el resto de sistemas.

- 2.6)

Comparar el tamaño de los ficheros de los datos de las tablas padron\_txt (txt), padron\_txt\_2 (txt pero no incluye los espacios innecesarios), padron\_parquet y padron\_parquet\_2 (alojados en hdfs cuya ruta se puede obtener de la propiedad location de cada tabla por ejemplo haciendo "show create table").

```
[cloudera@quickstart padron]$ hdfs dfs -du -s -h /user/hive/warehouse/*  
931.6 K  931.6 K  /user/hive/warehouse/padron_parquet  
2.1 K   2.1 K   /user/hive/warehouse/padron_parquet_2  
15.9 M  15.9 M  /user/hive/warehouse/padron_txt  
547     547     /user/hive/warehouse/padron_txt_2
```

### 3- Juguemos con Impala.

- 3.1)

¿Qué es Impala?

Es un motor SQL de elevado performance pensada para operar sobre grandes volúmenes de datos

- 3.2)

¿En qué se diferencia de Hive?

Es mucho más rápido que hive y está pensado para trabajar en tiempo real

- 3.3)

Comando INVALIDATE METADATA, ¿en qué consiste?

Si no se especifica ninguna tabla, elimina la cache de todas las tablas y la vuelve a sincronizar con el "Hive Metastore". Es una operación muy costosa a nivel de proceso, por lo que siempre que se pueda, se recomienda usar "Refresh".

- 3.4)

Hacer invalidate metadata en Impala de la base de datos datos\_padron.

- 3.5)

Calcular el total de EspanolesHombres, espanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC\_DISTrito y DESC\_BARRIO.

```
select p.desc_distrito, p.desc_barrio, sum(cast(p.espanoleshombres as int)) as "Hombres españoles",
sum(cast(p.espanolesmujeres as int)) as "Mujeres españolas",
sum(cast(p.extranjeroshombres as int)) as "Hombre extranjeros",
sum(cast(p.extranjerosmujeres as int)) as "Mujeres extranjeras"
from padron_parquet_2 as p group by p.desc_distrito, p.desc_barrio;
```

Query History Saved Queries Results (133)

	desc_distrito	desc_barrio	hombres españoles	mujeres españolas	hombre extranjeros	mujeres extranjeras
1	SAN BLAS-CANILLEJAS	EL SALVADOR	4887	5449	456	552
2	MONCLOA-ARAVACA	CIUDAD UNIVERSITARIA	6786	7858	636	853
3	HORTALEZA	CANILLAS	16937	19264	1760	2161
4	RETIRO	LOS JERONIMOS	2888	3228	334	448
5	HORTALEZA	PIOVERA	6013	6549	1066	1249
6	HORTALEZA	PALOMAS	3023	3117	349	419
7	MORATALAZ	HORCAJO	2872	3128	173	191
8	HORTALEZA	VALDEFUENTES	28327	28796	2807	3680

- 3.6)

Llevar a cabo las consultas en Hive en las tablas padron\_txt\_2 y padron\_parquet\_2 (No deberían incluir espacios innecesarios). ¿Alguna conclusión?

Parquet:

```
hive> select p.desc_distrito, p.desc_barrio, sum(cast(p.espanoleshombres as int)),
> sum(cast(p.espanolesmujeres as int)),
> sum(cast(p.extranjeroshombres as int)),
> sum(cast(p.extranjerosmujeres as int))
> from padron_parquet_2 as p group by p.desc_distrito, p.desc_barrio;
Query ID = cloudera_20210421140101_637d99a0-197d-4064-b009-515fbflab84b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

```
USERA    ORCASUR 5647    5971    1239    1357
USERA    PRADOLONGO 5051    6092    3273    3329
USERA    SAN FERMIN 9007    9895    2319    2534
USERA    ZOFIO 4830    5603    1829    1932
VICALVARO    CASCO H.VICALVARO 13381    14559    3729    3811
VICALVARO    EL CAJAVERAL 2441    2287    213    298
VICALVARO    VALDEBERNARDO 7720    8507    616    670
VICALVARO    VALDERRIVAS 8459    8821    295    379
VILLA DE VALLECAS    CASCO H.VALLECAS 15692    17097    3480    3696
VILLA DE VALLECAS    ENSANCHE DE VALLECAS 22032    22326    2807    3281
VILLA DE VALLECAS    SANTA EUGENIA 10395    11269    1238    1392
VILLAVERDE    BUTARQUE 8656    9127    1435    1580
VILLAVERDE    LOS ANGELES 12415    14279    2838    3128
VILLAVERDE    LOS ROSALES 14243    15676    3813    3878
VILLAVERDE    SAN CRISTOBAL 5166    5694    3168    2932
VILLAVERDE    VILLAVERDE ALTO C.H. 17089    19064    5026    5297
Time taken: 204.046 seconds, Fetched: 133 row(s)
```

TXT:



```
hive> select p.desc_distrito, p.desc_barrio, sum(cast(p.espanoleshombres as int)),
> sum(cast(p.espanolesmujeres as int)),
> sum(cast(p.extranjeroshombres as int)),
> sum(cast(p.extranjerosmujeres as int))
> from padron_txt_2 as p group by p.desc_distrito, p.desc_barrio;
Query ID = cloudera_20210421140606_9daeel13-b920-4f16-a4b8-5cfb7aeb8c5b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

```
TETUAN CASTILLEJOS 7839 9417 1348 1903
TETUAN CUATRO CAMINOS 13248 15998 2348 3012
TETUAN VALDEACEDERAS 9552 10682 2827 3472
USERA ALMENDRALES 7378 8706 3157 3238
USERA MOSCARDO 8502 10252 3810 3838
USERA ORCASITAS 9811 10997 1094 1224
USERA ORCASUR 5647 5971 1239 1357
USERA PRADOLONGO 5051 6092 3273 3329
USERA SAN FERMIN 9007 9895 2319 2534
USERA ZOFIO 4830 5603 1829 1932
VICALVARO CASCO H.VICALVARO 13381 14559 3729 3811
VICALVARO EL CAÑAVERAL 2441 2287 213 298
VICALVARO VALDEBERNARDO 7720 8507 616 670
VICALVARO VALDERRIVAS 8459 8821 295 379
VILLA DE VALLECAS CASCO H.VALLECAS 15692 17097 3480 3696
VILLA DE VALLECAS ENSANCHE DE VALLECAS 22032 22326 2807 3281
VILLA DE VALLECAS SANTA EUGENIA 10395 11269 1238 1392
VILLASVERDE BUTARQUE 8656 9127 1435 1580
VILLASVERDE LOS ANGELES 12415 14279 2838 3128
VILLASVERDE LOS ROSALES 14243 15676 3813 3878
VILLASVERDE SAN CRISTOBAL 5166 5694 3168 2932
VILLASVERDE VILLASVERDE ALTO C.H. 17089 19064 5026 5297
Time taken: 154.368 seconds, Fetched: 132 row(s)
```

En este caso, el formato texto ha sido más rápido

- 3.7)

Llevar a cabo la misma consulta sobre las mismas tablas en Impala. ¿Alguna conclusión?

Impala es muchísimo más rápido

- 3.8)

¿Se percibe alguna diferencia de rendimiento entre Hive e Impala?

Sin duda, Impala es tremendamente más rápido

#### 4- Sobre tablas particionadas.

- 4.1)

Crear tabla (Hive) padron\_particionado particionada por campos DESC\_DISTrito y DESC\_BARRIO cuyos datos estén en formato parquet.



```

hive> create table padron_particionado (
  > cod_distrito int,
  > cod_dist_barrio int,
  > cod_barrio int,
  > cod_dist_seccion int,
  > cod_seccion int,
  > cod_edad_int int,
  > espanoleshombres int,
  > espanolesmujeres int,
  > extranjeroshombres int,
  > extranjerosmujeres int)
  > partitioned by (desc_distrito string, desc_barrio string)
  > row format delimited
  > fields terminated by '\;'
  > stored as parquet
  > tblproperties("skip.header.line.count" = "1");
OK
Time taken: 0.334 seconds
hive> desc padron_particionado;
OK
cod_distrito          int
cod_dist_barrio       int
cod_barrio            int
cod_dist_seccion      int
cod_seccion           int
cod_edad_int          int
espanoleshombres      int
espanolesmujeres      int
extranjeroshombres    int
extranjerosmujeres    int
desc_distrito         string
desc_barrio           string

# Partition Information
# col_name            data_type            comment

desc_distrito        string
desc_barrio           string
Time taken: 0.575 seconds, Fetched: 18 row(s)

```

- 4.2)

Insertar datos (en cada partición) dinámicamente (con Hive) en la tabla recién creada a partir de un select de la tabla padron\_parquet\_2.

```

hive> SET hive.exec.dynamic.partitions = true;
hive> SET hive.exec.dynamic.partitions.maxpartitions = 10000;
hive> SET mapreduce.map.memory.mb = 2048;
hive> SET mapreduce.reduce.memory.mb = 2048;
hive> SET mapreduce.map.java.opts=-Xmx1800m;

```

```

hive> SET hive.exec.max.dynamic.partitions = 10000;
hive> SET hive.exec.max.dynamic.partitions.pernode = 1000;
hive> SET mapreduce.map.memory.mb = 2048;
hive> SET mapreduce.reduce.memory.mb = 2048;
hive> SET mapreduce.map.java.opts=-Xmx1800m;

```

```
hive> from padron_parquet_2
>
> insert overwrite table padron_particionado
>
> partition(desc_distrito, desc_barrio)
>
> select cod_distrito, cod_dist_barrio, cod_barrio,
> cod_dist_seccion , cod_seccion , cod_edad_int ,
> espanoleshombres , espanolesmujeres , extranjeroshombres ,
> extranjerosmujeres , desc_distrito, desc_barrio;
Query ID = cloudera_20210422090707_7a1468ca-79a9-4844-855c-4720a271adb0
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
```

- 4.3)

Hacer invalidate metadata en Impala de la base de datos padron\_particionado.



- 4.4)

Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC\_DISTrito y DESC\_BARRIO para los distritos CENTRO, LATINA, CHAMARTIN, TETUAN, VICALVARO y BARAJAS.

```
select desc_distrito, desc_barrio, sum(espanoleshombres),
sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)
from padron_particionado
where desc_distrito="CENTRO" || desc_distrito="LATINA" || desc_distrito="CHAMARTIN" ||
desc_distrito="TETUAN" || desc_distrito="VICALVARO" || desc_distrito="BARAJAS"
group by desc_distrito, desc_barrio;
```

Query History Saved Queries Results (11)

	desc_distrito	desc_barrio	sum(espanoleshombres)	sum(espanolesmujeres)	sum(extranjeroshombres)	sum(extranjerosmujeres)
1	TETUAN	VALDEACEDERAS	9552	10682	2827	3472
2	BARAJAS	CASCO H.BARAJAS	2948	3088	630	824
3	TETUAN	ALMENARA	8852	10176	1626	2048
4	TETUAN	BERRUGUETE	8743	10507	2672	3534
5	CENTRO	PALACIO	9346	9656	2364	2317
6	BARAJAS	CORRALEJOS	3558	3604	221	282
7	LATINA	LOS CARMENES	6884	7721	1427	1552
8	LATINA	PUERTA DEL ANGEL	15180	17760	4110	4709
9	CHAMARTIN	CIUDAD JARDIN	7305	8986	997	1312
10	LATINA	LAS AGUILAS	20862	24077	2981	3527
11	TETUAN	BELLAS VISTAS	10213	12003	3140	3846

- 4.5)

Llevar a cabo la consulta en Hive en las tablas padron\_parquet y padron\_partitionado. ¿Alguna conclusión?

Padron\_partitionada

```
hive> select desc_distrito, desc_barrio, sum(espanoleshombres),
> sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)
> from padron_particionado
> where desc_distrito="CENTRO" OR desc_distrito="LATINA" OR desc_distrito="CHAMARTIN" OR
> desc_distrito="TETUAN" OR desc_distrito="VICALVARO" OR desc_distrito="BARAJAS"
> group by desc_distrito, desc_barrio;
Query ID = cloudera_20210422093737_31e47bc3-c157-40f1-94e2-836baa3194a0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

```
BARAJAS CASCO H.BARAJAS 2948 3088 630 824
BARAJAS CORRALEJOS 3558 3604 221 282
CENTRO PALACIO 9346 9656 2364 2317
CHAMARTIN CIUDAD JARDIN 7305 8986 997 1312
LATINA LAS AGUILAS 20862 24077 2981 3527
LATINA LOS CARMENES 6884 7721 1427 1552
LATINA PUERTA DEL ANGEL 15180 17760 4110 4709
TETUAN ALMENARA 8852 10176 1626 2048
TETUAN BELLAS VISTAS 10213 12003 3140 3846
TETUAN BERRUGUETE 8743 10507 2672 3534
TETUAN VALDEACEDERAS 9552 10682 2827 3472
Time taken: 205.088 seconds, Fetched: 11 row(s)
```

Padron\_parquet\_2

```
hive> select desc_distrito, desc_barrio, sum(espanoleshombres),
> sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)
> from padron_parquet_2
> where desc_distrito="CENTRO" OR desc_distrito="LATINA" OR desc_distrito="CHAMARTIN" OR
> desc_distrito="TETUAN" OR desc_distrito="VICALVARO" OR desc_distrito="BARAJAS"
> group by desc_distrito, desc_barrio;
Query ID = cloudera_20210422094343_64d71531-a34b-44b4-b22e-52028b878181
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

```
VICALVARO      CASCO H.VICALVARO      13381.0 14559.0 3729.0 3811.0
VICALVARO      EL CAJAVERAL          2441.0 2287.0 213.0 298.0
VICALVARO      VALDEBERNARDO        7720.0 8507.0 616.0 670.0
VICALVARO      VALDERRIVAS          8459.0 8821.0 295.0 379.0
Time taken: 237.063 seconds, Fetched: 35 row(s)
```

La tabla particionada es más rápida y en la tabla parquet aparecen más resultados

- 4.6)

Llevar a cabo la consulta en Impala en las tablas padron\_parquet y padron\_particionado. ¿Alguna conclusión?

Aparecen los mismos resultados que en Hive

- 4.7)

Hacer consultas de agregación (Max, Min, Avg, Count) tal cual el ejemplo anterior con las 3 tablas (padron\_txt\_2, padron\_parquet\_2 y padron\_particionado) y comparar rendimientos tanto en Hive como en Impala y sacar conclusiones.

## 5- Trabajando con tablas en HDFS.

A continuación vamos a hacer una inspección de las tablas, tanto externas (no gestionadas) como internas (gestionadas). Este apartado se hará si se tiene acceso y conocimiento previo sobre cómo insertar datos en HDFS.

- 5.1)

Crear un documento de texto en el almacenamiento local que contenga una secuencia de números distribuidos en filas y separados por columnas, llámalo datos1 y que sea por ejemplo:

1,2,3

4,5,6

7,8,9

- 5.2)

Crear un segundo documento (datos2) con otros números pero la misma estructura.

```
[cloudera@quickstart ej5]$ cat datos1
1,2,3
4,5,6
7,8,9
[cloudera@quickstart ej5]$ cat datos2
5,3,8
2,1,6
7,6,4
```

- 5.3)

Crear un directorio en HDFS con un nombre a placer, por ejemplo, /test. Si estás en una máquina Cloudera tienes que asegurarte de que el servicio HDFS está activo ya que puede no iniciarse al encender la máquina (puedes hacerlo desde el Cloudera Manager). A su vez, en las máquinas Cloudera es posible (dependiendo de si usamos Hive desde consola o desde Hue) que no tengamos permisos para crear directorios en HDFS salvo en el directorio /user/cloudera.

```
[cloudera@quickstart ej5]$ hadoop fs -mkdir /user/cloudera/test
```

- 5.4)

Mueve tu fichero datos1 al directorio que has creado en HDFS con un comando desde consola.

```
[cloudera@quickstart ej5]$ hadoop fs -put /home/cloudera/ej5/datos1 /user/cloudera/test
```

- 5.5)

Desde Hive, crea una nueva database por ejemplo con el nombre numeros. Crea una tabla que no sea externa y sin argumento location con tres columnas numéricas, campos separados por coma y delimitada por filas. La llamaremos por ejemplo numeros\_tbl.

```
hive> create database numeros;
OK
Time taken: 1.094 seconds
```

```
hive> create table numeros_tbl (
> num1 int,
> num2 int,
> num3 int
> )
> row format delimited
> fields terminated by ',';
OK
Time taken: 4.077 seconds
```

- 5.6)

Carga los datos de nuestro fichero de texto datos1 almacenado en HDFS en la tabla de Hive. Consulta la localización donde estaban anteriormente los datos almacenados. ¿Siguen estando ahí? ¿Dónde están?. Borra la tabla, ¿qué ocurre con los datos almacenados en HDFS?

```
hive> load data inpath '/user/cloudera/test/datos1' into table numeros_tbl;
Loading data to table numeros.numeros_tbl
Table numeros.numeros_tbl stats: [numFiles=1, totalSize=18]
OK
Time taken: 0.76 seconds
```

```
hive> drop table numeros_tbl;
OK
Time taken: 0.194 seconds
```

Ya no hay nada en la carpeta de hdfs, al cargarse en la tabla, se ha borrado. Al borrar la tabla, los datos se han borrado también.

- 5.7)

Vuelve a mover el fichero de texto datos1 desde el almacenamiento local al directorio anterior en HDFS.

```
[cloudera@quickstart ej5]$ hadoop fs -put datos1 /user/cloudera/test
```

- 5.8)

Desde Hive, crea una tabla externa sin el argumento location. Y carga datos1 (desde HDFS) en ella. ¿A dónde han ido los datos en HDFS?

Borra la tabla ¿Qué ocurre con los datos en hdfs?

```
hive> create external table numeros_tbl (
> num1 int,
> num2 int,
> num3 int
> )
> row format delimited
> fields terminated by ',';
OK
Time taken: 0.243 seconds
```

```
hive> load data inpath '/user/cloudera/test/datos1' into table numeros_tbl;
Loading data to table numeros.numeros_tbl
Table numeros.numeros_tbl stats: [numFiles=1, totalSize=18]
OK
Time taken: 1.184 seconds
```

```
[cloudera@quickstart ej5]$ hadoop fs -ls /user/cloudera/test/
[cloudera@quickstart ej5]$
```

Ha desaparecido

```
hive> drop table numeros_tbl;
OK
Time taken: 0.254 seconds
hive>
```

- 5.9)

Borra el fichero datos1 del directorio en el que estén. Vuelve a insertarlos en el directorio que creamos inicialmente (/test). Vuelve a crear la tabla numeros desde hive pero ahora de manera externa y con un argumento location que haga referencia al directorio donde los

hayas situado en HDFS (/test). No cargues los datos de ninguna manera explícita. Haz una consulta sobre la tabla que acabamos de crear que muestre todos los registros. ¿Tiene algún contenido?

```
hive> create external table numeros_tbl (
  > num1 int,
  > num2 int,
  > num3 int
  > )
  > row format delimited
  > fields terminated by ','
  > location '/user/cloudera/test/';
OK
Time taken: 0.111 seconds
```

```
hive> select * from numeros_tbl;
OK
1      2      3
4      5      6
7      8      9
Time taken: 0.18 seconds, Fetched: 3 row(s)
```

Sí, está el contenido del fichero datos1

- 5.10)

Inserta el fichero de datos creado al principio, "datos2" en el mismo directorio de HDFS que "datos1". Vuelve a hacer la consulta anterior sobre la misma tabla. ¿Qué salida muestra?

```
[cloudera@quickstart ej5]$ hadoop fs -put datos2 /user/cloudera/test
```

```
hive> select * from numeros_tbl;
OK
1      2      3
4      5      6
7      8      9
5      3      8
2      1      6
7      6      4
Time taken: 0.197 seconds, Fetched: 6 row(s)
```

Muestra los dos ficheros en la tabla

- 5.11)

Extrae conclusiones de todos estos anteriores apartados.

6- Un poquito de Spark.



La siguiente sección de la práctica se abordará si ya se tienen suficientes conocimientos de Spark, en concreto de el manejo de DataFrames, y el manejo de tablas de Hive a través de Spark.sql.

- 6.1)

Comenzamos realizando la misma práctica que hicimos en Hive en Spark, importando el csv. Sería recomendable intentarlo con opciones que quiten las "" de los campos, que ignoren los espacios innecesarios en los campos, que sustituyan los valores vacíos por 0 y que infiera el esquema.

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

val spark = SparkSession
  .builder
  .appName("Padron")
  .getOrCreate()
```

Intitializing Scala interpreter ...

Spark Web UI available at <http://EM2021002738.bosonit.local:4040>

SparkContext available as 'sc' (version = 3.1.1, master = local[\*], app id = local-161

SparkSession available as 'spark'

```
import org.apache.spark.sql.SparkSession
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@74e6d80e
```

```
val file = "./padron/Rango_Edades_Seccion_202104.csv"
val schema = "COD_DISTrito int, DESC_DISTrito string, COD_DIST_BARRIO int, DESC_BARRIO string, " +
"COD_BARRIO int, COD_DIST_SECCION int, COD_SECCION int, COD_EDAD_INT int, EspanolesHombres int, " +
"EspanolesMujeres int, ExtranjerosHombres int, ExtranjerosMujeres int"
```

```
file: String = ./padron/Rango_Edades_Seccion_202104.csv
schema: String = COD_DISTrito int, DESC_DISTrito string, COD_DIST_BARRIO int, DESC_BARRIO string, COD_DISTrito int, COD_SECCION int, COD_EDAD_INT int, EspanolesHombres int, EspanolesMujeres int, ExtranjerosHombres int, ExtranjerosMujeres int
```

```
val df = spark.read.format("csv")
  .option("header", "true")
  .schema(schema)
  .option("sep", ";")
  .load(file)|
  .na.fill(0)
```

```
df: org.apache.spark.sql.DataFrame = [COD_DISTrito, DESC_DISTrito, COD_DIST_BARRIO, DESC_BARRIO, COD_DISTrito, COD_SECCION, COD_EDAD_INT, EspanolesHombres, EspanolesMujeres, ExtranjerosHombres, ExtranjerosMujeres]
```

```
println(df.printSchema)
```

```
root
|-- COD_DISTrito: integer (nullable = false)
|-- DESC_DISTrito: string (nullable = true)
|-- COD_DIST_BARRIO: integer (nullable = false)
|-- DESC_BARRIO: string (nullable = true)
|-- COD_BARRIO: integer (nullable = false)
|-- COD_DIST_SECCION: integer (nullable = false)
|-- COD_SECCION: integer (nullable = false)
|-- COD_EDAD_INT: integer (nullable = false)
|-- EspanolesHombres: integer (nullable = false)
|-- EspanolesMujeres: integer (nullable = false)
|-- ExtranjerosHombres: integer (nullable = false)
|-- ExtranjerosMujeres: integer (nullable = false)
```

```
df.select("ExtranjerosHombres").show(5)
```

```
+-----+
|ExtranjerosHombres|
+-----+
|                  0|
|                  0|
|                  0|
|                  0|
|                  0|
+-----+
only showing top 5 rows
```

```
df.show(5)
```

```
+-----+-----+-----+-----+-----+
|COD_DISTrito|DESC_DISTrito|COD_DIST_BARRIO|DESC_BARRIO|COD_BARRIO|C
panolesHombres|EspanolesMujeres|ExtranjerosHombres|ExtranjerosMujeres|
+-----+-----+-----+-----+-----+
|          1|CENTRO|          101|PALACIO|          1|
1|          0|          0|          0|          1|
|          1|CENTRO|          101|PALACIO|          1|
1|          1|          0|          0|          1|
|          1|CENTRO|          101|PALACIO|          1|
0|          1|          0|          0|          1|
|          1|CENTRO|          101|PALACIO|          1|
```

- 6.2)

De manera alternativa también se puede importar el csv con menos tratamiento en la importación y hacer todas las modificaciones para alcanzar el mismo estado de limpieza de los datos con funciones de Spark.

- 6.3)

Enumera todos los barrios diferentes.

```
df.select("desc_barrio").distinct.show(500, false)
```

```
+-----+
|desc_barrio|
+-----+
|EMBAJADORES|
|SANTA EUGENIA|
|LEGAZPI|
|DELICIAS|
|CIUDAD UNIVERSITARIA|
|CONCEPCION|
|VENTAS|
|IMPERIAL|
|SAN FERMIN|
|TRAFALGAR|
|ESTRELLA|
|PIUFRTA RONTTA|
```

- 6.4)

Crea una vista temporal de nombre "padron" y a través de ella cuenta el número de barrios diferentes que hay.

```
df.createOrReplaceTempView("padron")
```

```
spark.sql("""select count(distinct desc_barrio) from padron""").show()
```

```
+-----+
|count(DISTINCT desc_barrio)|
+-----+
|132|
+-----+
```

- 6.5)

Crea una nueva columna que muestre la longitud de los campos de la columna DESC\_DISTRITO y que se llame "longitud".

```
df.withColumn("longitud", length(col("desc_distrito"))).show()
```

```
+-----+-----+-----+-----+-----+-----+
|COD_DISTRITO|DESC_DISTRITO|COD_DIST_BARRIO|DESC_BARRIO|COD_BARRIO|COD_DIST_
panolesHombres|EspanolesMujeres|ExtranjerosHombres|ExtranjerosMujeres|longitud|
+-----+-----+-----+-----+-----+-----+
|1|CENTRO|101|PALACIO|1|1|
|0|0|20|1|
|1|CENTRO|101|PALACIO|1|1|
|1|1|0|20|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|0|20|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|0|20|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|0|20|1|
|1|CENTRO|101|PALACIO|1|1|
|3|1|0|2|20|1|
```

- 6.6)

Crea una nueva columna que muestre el valor 5 para cada uno de los registros de la tabla.

```
val df3 = df.withColumn("Valor 5", lit("5"))
```

```
df3.show()
```

```
+-----+-----+-----+-----+-----+-----+
|COD_DISTRITO|DESC_DISTRITO|COD_DIST_BARRIO|DESC_BARRIO|COD_BARRIO|C
panolesHombres|EspanolesMujeres|ExtranjerosHombres|ExtranjerosMujeres|Valor 5|
+-----+-----+-----+-----+-----+-----+
|1|CENTRO|101|PALACIO|1|1|
|1|0|5|1|
|1|CENTRO|101|PALACIO|1|1|
|1|1|5|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|5|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|5|1|
|1|CENTRO|101|PALACIO|1|1|
|0|1|5|1|
|1|CENTRO|101|PALACIO|1|1|
|3|1|5|1|
|1|CENTRO|101|PALACIO|1|1|
|2|2|5|1|
|1|CENTRO|101|PALACIO|1|1|
```

- 6.7)

Borra esta columna.

- 6.8)

Particiona el DataFrame por las variables DESC\_DISTRITO y DESC\_BARRIO.

```
df.write.partitionBy("DESC_DISTRITO", "DESC_BARRIO").mode("overwrite").saveAsTable("partitioned")
```

```
df.write.format("csv").partitionBy("DESC_DISTRITO", "DESC_BARRIO").mode("overwrite")  
  .save("./particion")
```

- 6.9)

Almacénalo en caché. Consulta en el puerto 4040 (UI de Spark) de tu usuario local el estado de los rdds almacenados.

```
val file_part = "./particion"  
  
val df_part = spark.read.format("csv")  
  .option("header", "true")  
  .schema(schema)  
  .option("sep", ";")  
  .load(file_part)  
  .na.fill(0)
```

```
df_part.cache()
```

```
res63: df_part.type = [COD_DISTRITO: int, COD_DIST_BARRIO: int ... 10 more fields]
```

- 6.10)

Lanza una consulta contra el DF resultante en la que muestre el número total de "espanoleshombres", "espanolesmujeres", "extranjeroshombres" y "extranjerosmujeres" para cada barrio de cada distrito. Las columnas distrito y barrio deben ser las primeras en aparecer en el show. Los resultados deben estar ordenados en orden de más a menos según la columna "extranjerosmujeres" y desempatarán por la columna "extranjeroshombres".

```
df_part.groupBy("desc_distrito", "desc_barrio")  
  .agg(sum("espanoleshombres"), sum("espanolesmujeres"),  
    sum("extranjeroshombres"), sum("extranjerosmujeres"))  
  .orderBy("sum(extranjerosmujeres)", "sum(extranjeroshombres)")  
  .show()
```

- 6.11)

Elimina el registro en caché.

```
df_part.unpersist()
```

```
res73: df_part.type = [COD_DISTrito: int, COD_DIST_BARRIO: int ... 10 more fields]
```

- 6.12)

Crea un nuevo DataFrame a partir del original que muestre únicamente una columna con DESC\_BARRIO, otra con DESC\_DISTrito y otra con el número total de "espanoleshombres" residentes en cada distrito de cada barrio. Únelo (con un join) con el DataFrame original a través de las columnas en común.

```
val nuevoDF = df.groupBy("desc_distrito", "desc_barrio")
                .agg(sum("espanoleshombres") as "espanoleshombres")
```

```
nuevoDF: org.apache.spark.sql.DataFrame = [desc_distrito: string, desc
```

```
nuevoDF.show()
```

```
+-----+-----+-----+
| desc_distrito | desc_barrio | espanoleshombres |
+-----+-----+-----+
| TETUAN        | VALDEACEDERAS | 9552 |
| CHAMARTIN     | CIUDAD JARDIN  | 7305 |
| ARGANZUELA    | DELICIAS       | 11743 |
| MORATALAZ     | VINATERO      | 6470 |
| CARABANCHEL   | VISTA ALEGRE   | 16016 |
| SAN BLAS-CANILLEJAS | SIMANCAS    | 10737 |
| CARABANCHEL   | PUERTA BOMITA  | 12555 |
```

```
nuevoDF.join(df, df("desc_distrito")==nuevoDF("desc_distrito") && df("desc_barrio")==nuevoDF("desc_barrio")).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| desc_distrito | desc_barrio | espanoleshombres | COD_DISTrito | DESC_DISTrito | COD_DIST_BARRIO | DESC_BAR |
RIO | COD_BARRIO | COD_DIST_SECCION | COD_SECCION | COD_EDAD_INT | EspanolesHombres | EspanolesMujeres | ExtranjerosHombres | ExtranjerosMujere |
s |
+-----+-----+-----+-----+-----+-----+-----+-----+
| TETUAN        | VALDEACEDERAS | 9552 | 6 | TETUAN        | 605 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 3 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 6 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 1 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 6 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 2 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 3 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 1 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 6 | TETUAN        | 0 | VALDEACEDERAS |
| TETUAN        | VALDEACEDERAS | 9552 | 0 | TETUAN        | 0 | VALDEACEDERAS |
```

- 6.13)

Repite la función anterior utilizando funciones de ventana. (over(Window.partitionBy.....)).

```
spark.sql("""DROP TABLE IF EXISTS padron_tbl;""")
```

```
res81: org.apache.spark.sql.DataFrame = []
```

```
spark.sql("""
create table padron_tbl as
select desc_distrito, desc_barrio, sum(espanoleshombres)
from padron
group by desc_distrito, desc_barrio;
""").show()
```

- 6.14)

Mediante una función Pivot muestra una tabla (que va a ser una tabla de contingencia) que contenga los valores totales (la suma de valores) de espanolesmujeres para cada distrito y en cada rango de edad (COD\_EDAD\_INT). Los distritos incluidos deben ser únicamente CENTRO, BARAJAS y RETIRO y deben figurar como columnas. El aspecto debe ser similar a este:

```
val nuevoDF = df.filter(col("desc_distrito").contains("BARAJAS") ||
                        col("desc_distrito").contains("CENTRO") ||
                        col("desc_distrito").contains("RETIRO"))
                .groupBy("desc_distrito", "cod_edad_int")
                .pivot("desc_distrito")
                .agg(sum("espanolesmujeres") as "espanolesmujeres")
                .na.fill(0)
```

```
nuevoDF.show()
```

desc_distrito	cod_edad_int	BARAJAS	CENTRO	RETIRO
BARAJAS	42	400	0	0
BARAJAS	98	12	0	0
CENTRO	34	0	883	0
BARAJAS	39	323	0	0
CENTRO	41	0	885	0
RETIRO	30	0	0	598
CENTRO	105	0	1	0
RETIRO	85	0	0	535
RETIRO	104	0	0	5
CENTRO	25	0	547	0
RETIRO	44	0	0	834
BARAJAS	95	20	0	0
BARAJAS	81	159	0	0

- 6.15)

Utilizando este nuevo DF, crea 3 columnas nuevas que hagan referencia a qué porcentaje de la suma de "espanolesmujeres" en los tres distritos para cada rango de edad representa cada uno de los tres distritos. Debe estar redondeada a 2 decimales. Puedes imponerte la condición extra de no apoyarte en ninguna columna auxiliar creada para el caso.



```
val pivotDF2 = pivotDF.withColumn("porcentajeBarajas", col("BARAJAS") /
  (col("BARAJAS") + col("CENTRO") + col("RETIRO")) * 100)
.withColumn("porcentajeCentro", col("CENTRO") / (col("BARAJAS") + col("CENTRO") + col("RETIRO")) * 100)
.withColumn("porcentajeRetiro", col("RETIRO") / (col("BARAJAS") + col("CENTRO") + col("RETIRO")) * 100)

pivotDF2.show()
```

cod_edad_int	BARAJAS	CENTRO	RETIRO	porcentajeBarajas	porcentajeCentro	porcentajeRetiro
31	191	931	615	10.995970063327576	53.59815774323546	35.40587219
85	95	354	535	9.654471544715447	35.97560975609756	54.36991869
65	212	656	951	11.654755360087961	36.063771302913686	52.28147333
53	342	754	788	18.152866242038215	40.02123142250531	41.825902335
78	187	451	654	14.473684210526317	34.907120743034056	50.61919504

- 6.16)

Guarda el archivo csv original particionado por distrito y por barrio (en ese orden) en un directorio local. Consulta el directorio para ver la estructura de los ficheros y comprueba que es la esperada.

```
df.write.format("csv").partitionBy("DESC_DISTrito", "DESC_BARRIO").mode("overwrite")
.save("./particion")
```


DESC_DISTrito=Ciudad Real-EL%20PA...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=FUENCARRAL-EL%20PA...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=HORTALEZA%20%20%2...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=LATINA%20%20%20%20...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=MONCLOA-ARAVACA%...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=MORATALAZ%20%20%2...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=PUENTE%20DE%20VALL...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=RETIRO%20%20%20%20...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=SALAMANCA%20%20%...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=SAN%20BLAS-CANILLEJ...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=TETUAN%20%20%20%2...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=USERA%20%20%20%20...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=VICALVARO%20%20%20...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=VILLA%20DE%20VALLEC...	22/04/2021 13:16	Carpeta de archivos	
DESC_DISTrito=VILLAVERDE%20%20%20...	22/04/2021 13:16	Carpeta de archivos	
._SUCCESS.crc	22/04/2021 13:16	Archivo CRC	1 KB
._SUCCESS	22/04/2021 13:16	Archivo	0 KB


Nombre	Fecha de modificación	Tipo
DESC_BARRIO=ARAVACA%20%20%20%2...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=ARGUELLES%20%20%20...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=CASA%20DE%20CAMPO...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=CIUDAD%20UNIVERSITAR...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=EL%20PLANTIO%20%20...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=VALDEMARIN%20%20%2...	22/04/2021 13:16	Carpeta de archivos
DESC_BARRIO=VALDEZARZA%20%20%2...	22/04/2021 13:16	Carpeta de archivos

- 6.17)

Haz el mismo guardado pero en formato parquet. Compara el peso del archivo con el resultado anterior.

```
df.write.format("parquet").partitionBy("DESC_DISTRITO", "DESC_BARRIO").mode("overwrite")
  .save("./particion2")
```

	particion
Tipo:	Carpeta de archivos
Ubicación:	C:\Users\diego.rodriguez
Tamaño:	6,89 MB (7.227.569 bytes)
Tamaño en disco:	7,25 MB (7.610.368 bytes)
Contiene:	324 archivos, 153 carpetas

	particion2
Tipo:	Carpeta de archivos
Ubicación:	C:\Users\diego.rodriguez
Tamaño:	1,16 MB (1.220.563 bytes)
Tamaño en disco:	1,50 MB (1.572.864 bytes)
Contiene:	324 archivos, 153 carpetas

El formato parquet ocupa un cuarto de lo que ocupa csv

#### 7- ¿Y si juntamos Spark y Hive?

- 7.1)

Por último, prueba a hacer los ejercicios sugeridos en la parte de Hive con el csv "Datos Padrón" (incluyendo la importación con Regex) **utilizando desde Spark EXCLUSIVAMENTE sentencias spark.sql**, es decir, **importar los archivos desde local directamente como tablas de Hive** y haciendo todas las consultas sobre estas tablas **sin transformarlas en ningún momento en DataFrames ni DataSets**.

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.hive.HiveContext

val spark = SparkSession
    .builder
    .appName("Padron")
    .getOrCreate()
|
```

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.hive.HiveContext
spark: org.apache.spark.sql.SparkSession = org.
```

```
spark.stop()
```

```
val spark = SparkSession
    .builder
    .appName("Padron")
    .enableHiveSupport()
    .getOrCreate()
```

```
spark: org.apache.spark.sql.SparkSession = org.
```

```
spark.sql("show tables").show()
```

database	tableName	isTemporary
default	numeros_tbl	false
default	padron_parquet	false
default	padron_parquet_2	false
default	padron_particionado	false
default	padron_txt	false
default	padron_txt_2	false

```
spark.sql("""create database diegodb""")
```

```
res3: org.apache.spark.sql.DataFrame = []
```

```
spark.sql("use diegodb")
```

```
res5: org.apache.spark.sql.DataFrame = []
```

```
spark.sql("""create table numeros_tbl (  
num1 int,  
num2 int,  
num3 int  
)  
row format delimited  
fields terminated by ',';""")
```

```
res7: org.apache.spark.sql.DataFrame = []
```

```
spark.sql("show tables").show()
```

```
+-----+-----+-----+  
|database|  tableName|isTemporary|  
+-----+-----+-----+  
| diegodb|numeros_tbl|      false|  
+-----+-----+-----+
```

```
spark.sql("""  
load data local inpath './datos1.txt' into table numeros_tbl  
""")
```

```
res11: org.apache.spark.sql.DataFrame = []
```

```
spark.sql("""select * from numeros_tbl""").show()
```

```
+-----+-----+-----+  
|num1|num2|num3|  
+-----+-----+-----+  
|   1|   2|   3|  
|   4|   5|   6|  
|   7|   8|   9|  
+-----+-----+-----+
```