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" .

```
CREATE DATABASE datos_padron;
```

• 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.

```
CREATE EXTERNAL TABLE padron_txt
(
COD_DISTRITO bigint ,
  DESC_DISTRITO string ,
  COD_DIST_BARRIO bigint ,
  DESC_BARRIO string ,
  COD_BARRIO bigint ,
  COD_DIST_SECCION bigint ,
  COD_SECCION bigint ,
  COD_EDAD_INT bigint ,
  EspanolesHombres bigint ,
  EspanolesMujeres bigint ,
  ExtranjerosHombres bigint ,
  ExtranjerosMujeres bigint
)

ROW FORMAT   SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
 WITH SERDEPROPERTIES ("separatorChar" = ";",
  "quoteChar"     = "\"",
  "escapeChar"    = "\\"
  )
STORED AS TextFile
LOCATION '/user/cloudera/Rango_Edades_Seccion_202110.csv_table'
TBLPROPERTIES("skip.header.line.count" = "1", 'transactional'='true');


load data local inpath '/home/cloudera/Desktop/Padron/Rango_Edades_Seccion_202110.csv' into
table padron_txt;
```

- 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.)

```
CREATE TABLE datos_padron.padron_txt_2
STORED AS TextFile
AS
SELECT COD_DISTRITO,
    TRIM(DESC_DISTRITO) as DESC_DISTRITO,
    COD_DIST_BARRIO,
    TRIM(DESC_BARRIO) as DESC_BARRIO,
    COD_BARRIO,
    COD_DIST_SECCION,
    COD_SECCION,
    COD_EDAD_INT,
    EspanolesHombres,
    EspanolesMujeres,
    ExtranjerosHombres,
    ExtranjerosMujeres
FROM datos_padron.padron_txt
```

- 1.4)
  Investigar y entender la diferencia de incluir la palabra LOCAL en el comando LOAD DATA.

El comando load data inpath se utiliza para cargar datos en la tabla de HIVE. 'LOCAL' significa que el archivo de entrada está en el sistema de archivos local. Si se omite 'LOCAL', busca el archivo en 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 utiliando 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.

```
CREATE TABLE datos_padron.padron_txt_3
STORED AS TextFile
AS
SELECT COD_DISTRITO,
	DESC_DISTRITO,
	COD_DIST_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 datos_padron.padron_txt



CREATE TABLE datos_padron.padron_txt_4
STORED AS TextFile
AS
SELECT COD_DISTRITO,
	DESC_DISTRITO,
	COD_DIST_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 datos_padron.padron_txt_2
```

- 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.

Tenemos 4 tablas:

padron_txt : tabla con espacios en blanco y celdas vacias
padron_txt_2 : tabla sin espacios en blanco y celdas vacias
padron_txt_3 : tabla con espacios en blanco y celdas con ceros
padron_txt_4 : tabla sin espacios en blanco y celdas con ceros

2- Investigamos el formato columnar parquet.

- 2.1) ¿Qué es CTAS?

Creación de una tabla a partir de los resultados de una consulta (CTAS)

https://docs.aws.amazon.com/es_es/athena/latest/ug/ctas.html

- 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.

```
CREATE TABLE datos_padron.padron_parquet
STORED AS parquet
AS
SELECT COD_DISTRITO,
    DESC_DISTRITO,
    COD_DIST_BARRIO,
    DESC_BARRIO,
    COD_BARRIO,
    COD_DIST_SECCION,
    COD_SECCION,
```

```
    COD_EDAD_INT,
    EspanolesHombres,
    EspanolesMujeres,
    ExtranjerosHombres,
    ExtranjerosMujeres
FROM datos_padron.padron_txt_3
```

- 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).

- 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.

```
CREATE TABLE datos_padron.padron_parquet_2
STORED AS parquet
AS
SELECT COD_DISTRITO,
    DESC_DISTRITO,
    COD_DIST_BARRIO,
    DESC_BARRIO,
    COD_BARRIO,
    COD_DIST_SECCION,
    COD_SECCION,
    COD_EDAD_INT,
    EspanolesHombres,
    EspanolesMujeres,
    ExtranjerosHombres,
    ExtranjerosMujeres
FROM datos_padron.padron_txt_4
```

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

https://programmerclick.com/article/39481908265/

- 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").

padron_txt_3 : 16,2 MB
padron_txt_4 : 11,9 MB
padron_parquet : 910,3 Kb
padron_parquet _2 : 908,3 Kb

3- Juguemos con Impala.

- 3.1) ¿Qué es Impala?

https://aprenderbigdata.com/apache-impala/

- 3.2) ¿En qué se diferencia de Hive?

https://programmerclick.com/article/6259840853/#:~:text=Hive%20es%20adecuado%20para%20el,experimenten%20y%20verifiquen%20ideas%20r%C3%A1pidamente.

- 3.3) Comando INVALIDATE METADATA, ¿en qué consiste?

https://impala.apache.org/docs/build/html/topics/impala_invalidate_metadata.html

- 3.4) Hacer invalidate metadata en Impala de la base de datos datos_padron.

```
INVALIDATE METADATA datos_padron.
```

- 3.5) Calcular el total de EspanolesHombres, espanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC_DISTRITO y DESC_BARRIO.
- 3.6) Llevar a cabo las consultas en Hive en las tablas padron_txt_4 y padron_parquet_2 (No deberían incluir espacios innecesarios). ¿Alguna conclusión?

```
select DESC_DISTRITO,
    DESC_BARRIO,
    sum(cast(EspanolesHombres as int)) as EspanolesHombres,
    sum(cast(espanolesMujeres as int)) as espanolesMujeres,
    sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
    sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.padron_parquet_2
group by DESC_DISTRITO,DESC_BARRIO;

select DESC_DISTRITO,
    DESC_BARRIO,
    sum(cast(EspanolesHombres as int)) as EspanolesHombres,
    sum(cast(espanolesMujeres as int)) as espanolesMujeres,
    sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
    sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.padron_txt_4
group by DESC_DISTRITO,DESC_BARRIO;
```

Entre Hive con txt o Hive con parquet no hay diferencia.

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

```
INVALIDATE METADATA datos_padron.padron_parquet_2;
select DESC_DISTRITO,
    DESC_BARRIO,
    sum(cast(EspanolesHombres as int)) as EspanolesHombres,
    sum(cast(espanolesMujeres as int)) as espanolesMujeres,
```

```
	sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
	sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.padron_parquet_2
group by DESC_DISTRITO,DESC_BARRIO;

INVALIDATE METADATA datos_padron.padron_txt_4;
select DESC_DISTRITO,
	DESC_BARRIO,
	sum(cast(EspanolesHombres as int)) as EspanolesHombres,
	sum(cast(espanolesMujeres as int)) as espanolesMujeres,
	sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
	sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.padron_txt_4
group by DESC_DISTRITO,DESC_BARRIO;
```

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

Impala es mucho mas rápido que HIVE, pero Hive muestra el resultado ordenado.
Entre Hive con txt o Hive con parquet no hay diferencia.

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.

```
CREATE EXTERNAL TABLE padron_particionado
(
COD_DISTRITO bigint ,
  COD_DIST_BARRIO bigint ,
  COD_BARRIO bigint ,
  COD_DIST_SECCION bigint ,
  COD_SECCION bigint ,
  COD_EDAD_INT bigint ,
  EspanolesHombres bigint ,
  EspanolesMujeres bigint ,
  ExtranjerosHombres bigint ,
  ExtranjerosMujeres bigint
)
PARTITIONED BY (DESC_DISTRITO string, DESC_BARRIO string)
STORED AS parquet
LOCATION '/user/cloudera/Rango_Edades_Seccion_202110.csv_table'
```

•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.

```
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
SET hive.auto.convert.join = false;
```

```
SET mapred.job.name =;
SET mapreduce.job.queuename = large;

SET hive.input.format = org.apache.hadoop.hive.ql.io.HiveInputFormat;
SET hive.support.sql11.reserved.keywords = false;
SET hive.script.operator.truncate.env = true;
SET hive.exec.dynamic.partition.mode = nonstrict;

SET hive.exec.parallel = true;
SET hive.auto.convert.join = false;
SET hive.exec.reducers.max = 128;
SET mapreduce.job.reduces = 1024;
SET mapreduce.job.running.map.limit = 100;
SET mapreduce.job.running.reduce.limit = 50;

SET mapred.child.java.opts = -Xmx2g -Xms512m;
SET mapreduce.map.memory.mb = 2048;
SET mapreduce.map.java.opts = -Xmx1638m -Dfile.encoding=UTF-8;
SET mapreduce.map.cpu.vcores = 1;
SET mapreduce.reduce.memory.mb = 2048;
SET mapreduce.reduce.java.opts = -Xmx1638m -Dfile.encoding=UTF-8;
SET mapreduce.reduce.cpu.vcores = 1;

insert overwrite table datos_padron.padron_particionado
partition(DESC_DISTRITO,DESC_BARRIO)
 select COD_DISTRITO,
  DESC_DISTRITO,
  COD_DIST_BARRIO,
  DESC_BARRIO,
  COD_BARRIO,
  COD_DIST_SECCION,
  COD_SECCION,
  COD_EDAD_INT,
  EspanolesHombres,
  EspanolesMujeres,
  ExtranjerosHombres,
  ExtranjerosMujeres
  from  datos_padron.padron_parquet_2;

insert overwrite table datos_padron.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 as DESC_DISTRITO,
   DESC_BARRIO as  DESC_BARRIO
  from  datos_padron.padron_parquet_2;
```

- 4.3) Hacer invalidate metadata en Impala de la base de datos padron_particionado.

```
INVALIDATE METADATA datos_padron;
```

- 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.

```
INVALIDATE METADATA datos_padron.padron_particionado;
select DESC_DISTRITO,
      DESC_BARRIO,
      sum(cast(EspanolesHombres as int)) as EspanolesHombres,
      sum(cast(espanolesMujeres as int)) as espanolesMujeres,
      sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
      sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.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;
```

3 seg

- 4.5) Llevar a cabo la consulta en Hive en las tablas padron_parquet y padron_partitionado. ¿Alguna conclusión?

```
select DESC_DISTRITO,
      DESC_BARRIO,
      sum(cast(EspanolesHombres as int)) as EspanolesHombres,
      sum(cast(espanolesMujeres as int)) as espanolesMujeres,
      sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
      sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.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;
```

19 seg

```
select DESC_DISTRITO,
      DESC_BARRIO,
      sum(cast(EspanolesHombres as int)) as EspanolesHombres,
      sum(cast(espanolesMujeres as int)) as espanolesMujeres,
      sum(cast(ExtranjerosHombres as int)) as ExtranjerosHombres,
      sum(cast(ExtranjerosMujeres as int)) as ExtranjerosMujeres
from datos_padron.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;
```

20 seg

Sigue siendo mucho más rápido Impala que hive.
Padron_particionado tarda lo mismo que padron_parquet

- 4.6) Llevar a cabo la consulta en Impala en las tablas padron_parquet y padron_particionado. ¿Alguna conclusión?

padron_parquet_2 : 2 seg
padron_parquet : 2,9
padron_particionado : 2,12

El mas rapido en parquet_2 que no contenia espacios.
Entre padron_particionado y parquet con espacios es más rápido particionado.

- 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.

```
select DESC_DISTRITO,
      DESC_BARRIO,
      max(espanoleshombres)as max_espanoles_hombres,
      min(espanoleshombres)as min_espanoles_hombres,
      avg(espanoleshombres)as avg_espanoles_hombres,
      count(espanoleshombres)as count_espanoles_hombres

from datos_padron.padron_parquet_2
group by DESC_DISTRITO,DESC_BARRIO;

INVALIDATE METADATA datos_padron.padron_parquet_2
select DESC_DISTRITO,
      DESC_BARRIO,
      max(espanoleshombres)as max_espanoles_hombres,
      min(espanoleshombres)as min_espanoles_hombres,
      avg(cast (espanoleshombres as int))as avg_espanoles_hombres,
      count(espanoleshombres)as count_espanoles_hombres
```

```
from datos_padron.padron_parquet_2
group by DESC_DISTRITO,DESC_BARRIO;
```

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.

- 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 HDhdfs

```
hdfs dfs -mkdir test
```

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

```
hdfs dfs -moveFromLocal datos1 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.

```
create database numeros;
CREATE TABLE numeros.numeros_tbl
(
  num1 bigint ,
  num2 bigint ,
  num3 bigint
 )

ROW FORMAT   SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
 WITH SERDEPROPERTIES ("separatorChar" = ",",
  "escapeChar"    = "\\"
  )
STORED AS TextFile
```

- 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? No ¿Dónde están?. Borra la tabla, ¿qué ocurre con los datos almacenados en HDFS? Ya no están

```
load data inpath '/datos1' into table numeros.numeros_tbl;
```

```
select * from numeros.numeros_tbl
```

	numeros_tbl.num1	numeros_tbl.num2	numeros_tbl.num3
1	1	2	3
2	4	5	6
3	7	8	9

```
[cloudera@quickstart ~]$ hdfs dfs -ls test
[cloudera@quickstart ~]$
```

```
drop table numeros.numeros_tbl
```

```
hdfs dfs -ls test
```

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

- 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? No estan

```
CREATE EXTERNAL TABLE numeros.numeros_tbl
(
  num1 bigint ,
  num2 bigint ,
  num3 bigint
 )
ROW FORMAT   SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
 WITH SERDEPROPERTIES ("separatorChar" = ",",
  "escapeChar"    = "\\"
  )
STORED AS TextFile
```

- 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?

```
CREATE EXTERNAL TABLE numeros.numeros_tbl
(
  num1 bigint ,
  num2 bigint ,
  num3 bigint
 )
ROW FORMAT   SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
 WITH SERDEPROPERTIES ("separatorChar" = ",",
  "escapeChar"    = "\\"
  )
STORED AS TextFile
LOCATION "/test"
```

• 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?

	numeros_tbl.num1 STRING_TYPE	numeros_tbl.num2	numeros_tbl.num3
1	1	2	3
2	4	5	6
3	7	8	9
4	16	12	5
5	4	8	9
6	12	14	7

• 5.11) Extrae conclusiones de todos estos anteriores apartados.

Cuando se crea una tabla externa se mantienen los datos en hdfs

6- Un poquito de Spark. **(EN NOTEBOOK)**

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.

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

• 6.5) Crea una nueva columna que muestre la longitud de los campos de la columna

DESC_DISTRITO y que se llame "longitud".

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

• 6.7) Borra esta columna.

• 6.8) Particiona el DataFrame por las variables DESC_DISTRITO y DESC_BARRIO.

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

• 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".

• 6.11) Elimina el registro en caché.

• 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.

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

• 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:

COD_EDAD_INT	BARAJAS	CENTRO	RETIRO
0	5.483870967741935	2.3545454545454545	3.4193548387096775
1	5.774193548387097	2.3423423423423424	3.9361702127659575
2	6.741935483870968	2.3394495412844036	4.258064516129032
3	7.580645161290323	2.2181818181818183	4.531914893617022
4	8.064516129032258	2.238532110091743	4.638297872340425
5	8.225806451612904	2.272727272727273	4.585106382978723
6	7.838709677419355	2.2818181818181817	4.76595744680851
7	8.709677419354838	2.128440366972477	4.585106382978723
8	8.129032258064516	2.390909090909091	4.659574468085107

• 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.

• 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.

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

7- ¿Y si juntamos Spark y Hive? (EN NOTEBOOK)

• 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.