Práctica Hive

1. 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. Crear Base de datos **datos\_padron**

créate database datos\_padron;

* 1. Crear tabla 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 carácter ';' y los campos estarán encerrados en comillas dobles '”’ y se deberá omitir la cabecera del fichero de datos al crear la tabla

create table padron\_txt(COD\_DISTRITO FLOAT, DESC\_DISTRITO STRING, COD\_DIST\_BARRIO FLOAT, DESC\_BARRIO STRING, COD\_BARRIO FLOAT, COD\_DIST\_SECCION FLOAT, COD\_SECCION FLOAT, COD\_EDAD\_INT FLOAT, EspanolesHombres FLOAT, EspanolesMujeres FLOAT, ExtranjerosHombres FLOAT, ExtranjerosMujeres FLOAT) row format serde 'com.bizo.hive.serde.csv.CSVSerde'

with serdeproperties (

"separatorChar" = "\073",

"quoteChar" = "'",

"escapeChar" = "\\"

)

TBLPROPERTIES("skip.header.line.count"="1");

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\073' TBLPROPERTIES("skip.header.line.count"="1");

load data inpath '/user/cloudera/practica/Rango\_Edades\_Seccion\_202104.csv' into table padron\_txt;

* 1. (OPCIONAL) Crear tabla padron\_txt2 que haga **trim** sobre los datos de algunas columnas con la finalidad de eliminar los espacios en blanco innecesarios.

create table padron\_txt2as select cod\_distrito, trim(desc\_distrito), cod\_dist\_barrio, trim(desc\_barrio), cod\_barrio, cod\_dist\_seccion, cod\_seccion, cod\_edad\_int, EspanolesHombres, EspanolesMujeres, ExtranjerosHombres, ExtranjerosMujeres from padrón\_txt;

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

Con local busca en el sistema de ficheros de la propia maquina si no busca en HDFS

* 1. 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 padraaa as select cod\_distrito, desc\_distrito, cod\_dist\_barrio, desc\_barrio, cod\_barrio, cod\_dist\_seccion, cod\_seccion, cod\_edad\_int, coalesce(EspanolesHombres,0), coalesce(EspanolesMujeres,0), coalesce(ExtranjerosHombres,0), coalesce(ExtranjerosMujeres,0) from padrón\_txt;

* 1. 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.

1. ¿Qué es CTAS?
   1. ¿Qué es CTAS?

create-table-as-select, una manera de crear tablas en hive basadas en una consulta

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

create table padron\_parquet stored as parquet as select \* from padron\_txt;

* 1. (OPCIONAL) Crear Tabla (en Hive) padron\_parquet2 (cuyos datos serán almacenados en el formato columnar **parquet**) a partir de la tabla padron\_txt2 mediante un CTAS.
  2. Investigar en que consiste el formato columnar parquet y las ventajas de trabajar con este tipo de formatos.
* Almacenamiento en columnas, puede realizar cálculos en todos los valores de una columna leyendo solo una fracción de datos
* Compresión flexible en distintos formatos, la compresión es visible para las aplicaciones que leen datos
* Permite trabajar con grandes volúmenes de datos
  1. Comparar el tamaño de los ficheros de los datos de las tablas padron\_txt (CSV) y padron\_parquet (alojados en hdfs cuya ruta se puede obtener de la propiedad location de cada tabla por ejemplo haciendo SHOW CREATE TABLE)

Show créate table padrón\_txt

hadoop fs -du /user/hive/warehouse/datos\_padron.db/

El de parquet pesa 7mb mientras que el de txt pesa 23mb

* 1. (OPCIONAL) Comparar el tamaño de los ficheros de los datos de las tablas padron\_txt (CSV), padron\_txt2, padron\_parquet y padron\_parquet2 (alojados en hdfs cuya ruta se puede obtener de la propiedad location de cada tabla por ejemplo haciendo SHOW CREATE TABLE)

1. Impala
   1. ¿Qué es impala?

Un motor SQL para operar sobre grandes volúmenes de datos

* 1. ¿En qué se diferencia a Hive?

Ejecuta las consultas directamente sobre el cluster lo que lo hace de entre 5 a 20 veces mas rápido, este tiene una latencia mucho mas baja y no puede usar indices

* 1. Comando INVALIDATE METADATA, ¿en qué consiste?

Marca los metadatos de una tabla como obsoletos para que en la siguiente consulta los recarge, si no se especifica una tabla lo hara para todas

* 1. Hacer invalidate metadata en Impala de Base de datos **datos\_padron**

Invalidate metadata

* 1. Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombresy ExtranjerosMujeres agrupado por DESC\_DISTRITO y DESC\_BARRIO.
     1. Llevar a cabo la consulta en Hive en las tablas padron\_txt y padron\_parquet. ¿Alguna conclusión?

select sum(coalesce(EspanolesHombres,0)), sum(coalesce(EspanolesMujeres,0)), sum(coalesce(ExtranjerosHombres,0)), sum(coalesce(ExtranjerosMujeres,0)) from padron\_txt group by DESC\_DISTRITO, DESC\_BARRIO;

39s sobre la txt y 43s sobre la parquet en realizar la operación, aunque en el tiempo de procesado de CPU tardo menos

* + 1. Llevar a cabo la consulta en Impala en las tablas padron\_txt y padron\_parquet. ¿Alguna conclusión?

Txt 6’5s mientras que parquet 7s, mucho mas rápido que hive e ira en aumento proporcionalmente a la cantidad de datos de las tablas

* + 1. ¿Se percibe alguna diferencia de rendimiento entre Hive e Impala?

Mas de medio minuto

1. Particionamiento
   1. Crear tabla (Hive) **padron\_particionado** particionada por los campos DESC\_DISTRITO y DESC\_BARRIO cuyos datos estén en formato parquet.

create table padron\_particionado(COD\_DISTRITO FLOAT, COD\_DIST\_BARRIO FLOAT, COD\_BARRIO FLOAT, COD\_DIST\_SECCION FLOAT, COD\_SECCION FLOAT, COD\_EDAD\_INT FLOAT, EspanolesHombres FLOAT, EspanolesMujeres FLOAT, ExtranjerosHombres FLOAT, ExtranjerosMujeres FLOAT) partitioned by (DESC\_DISTRITO STRING, DESC\_BARRIO STRING);

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

Hay que poner el modo no estricto

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 from padron\_txt2 limit 175210

* 1. Hacer invalidate metadata en Impala de Base de datos **padron\_particionado**
  2. Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombresy ExtranjerosMujeres agrupado por DESC\_DISTRITO y DESC\_BARRIO para los distritos CENTRO, LATINA, CHAMARTIN, TETUAN, VICALVARO y BARAJAS.
     1. Llevar a cabo la consulta en Hive en las tablas padron\_parquet y padron\_particionado. ¿Alguna conclusión?

select sum(coalesce(EspanolesHombres,0)), sum(coalesce(EspanolesMujeres,0)), sum(coalesce(ExtranjerosHombres,0)), sum(coalesce(ExtranjerosMujeres,0)) from padron\_parquet2 where DESC\_DISTRITO LIKE "CENTRO%" or DESC\_DISTRITO LIKE "LATINA%" or DESC\_DISTRITO LIKE "CHAMARTIN%" or DESC\_DISTRITO LIKE "TETUAN%" or DESC\_DISTRITO LIKE "VICALVARO%" or DESC\_DISTRITO LIKE "BARAJAS%" group by DESC\_DISTRITO, DESC\_BARRIO;

50s parquet mientras que la tabla particionada 56s y eso que solo tiene un cuarto de los datos

* + 1. Llevar a cabo la consulta en Impala en las tablas padron\_parquet y padron\_particionado. ¿Alguna conclusión?

6s parquet mientras que la tabla particionada 7’5s y eso que solo tiene un cuarto de los datos

Parquet es la mejor opción en cuanto a velocidad de procesado en cada prueba que se ha realizado

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

select max(coalesce(EspanolesHombres,0)) from padron\_particionado group by DESC\_DISTRITO, DESC\_BARRIO;

select min(coalesce(EspanolesMujeres,0)) from padron\_txt2 group by DESC\_DISTRITO, DESC\_BARRIO;

select avg(coalesce(ExtranjerosHombres,0)) from padron\_txt2 group by DESC\_DISTRITO, DESC\_BARRIO;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | h.parquet | h.particionado | h.txt | i.parquet | i.particionado | i.txt |
| min | 38 | 54 | 39 | 0’5 | 1 | 4 |
| max | 45 | 56 | 46 | 0’5 | 0’6 | 2’5 |
| avg | 43 | 57 | 43 | 0’5 | 0’6 | 0’65 |

Parquet es de lejos el mas rápido, haciendo todas las consultas en impala en alrededor de medio segundo y sacándole tiempo al particionado y eso que esta solo tiene un cuarto de los datos en mí caso, txt tiene algunas consultas rapidas pero en general siempre tarda bastante mas, ahora bien en hive las consultas en parquet y txt se igualan siendo solo superadas en 1 segundo la segundas mientras que las particionadas (teniendo solo 1 tercio de los datos) es del orden de 10 s mas lento

Información:

* <https://cwiki.apache.org/confluence/collector/pages.action?key=Hive>
* <https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-CreatingHiveTables>
* <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>
* <https://cwiki.apache.org/confluence/display/Hive/DynamicPartitions>

5- Trabajando con tablas en HDFS.

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

1. Crear un segundo documento (datos2) con otros números pero la misma estructura.
2. 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.
3. Mueve tu fichero datos1 al directorio que has creado en HDFS con un comando desde consola.
4. 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.
5. 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?

Créate database números;

Use números;

Créate table números\_tbl(m1 int, n2 int, n3 int) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’

Load data inpath ‘/test/datos1’ into table números\_tbl;

No están ahí, dentro de /user/hive/warehouse/números.db, se borran

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

Hadoop fs -put /x / test

1. 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?

Créate external table numero\_tbl2(n1 int, n2 int, n3 int) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;

A /user/hive/warehouse/números.db, Se mantienen

1. 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?

Haddop fs -rm -r

Créate external table numero\_tbl2(n1 int, n2 int, n3 int) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’ location ‘/user/test’;

Los números de datos 1

1. 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?

Ahora también muestra los números de datos 2

1. Extrae conclusiones de todos estos anteriores apartados.

Si almacenamos las tablas de manera interna tenemos podemos borrar los datos ya que somos los administradores, sim embargo si es externa al borrar la tabla solo eliminaremos los metadatos mientras que la tabla en si permanecerá , al indicarle la localización tenemos la ventaja de que con mover los archivos ya hace las inserciones si el archivo está bien construido cosa que parece la mejor opción

Spark

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.
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 delos datos con funciones de Spark.
3. Enumera todos los barrios diferentes.
4. Crea una vista temporal de nombre "padron" y a través de ella cuenta el número de barriosdiferentes que hay.
5. Crea una nueva columna que muestre la longitud de los campos de la columna DESC\_DISTRITO y que se llame "longitud".
6. Crea una nueva columna que muestre el valor 5 para cada uno de los registros de la tabla.
7. Borra esta columna.
8. Particiona el DataFrame por las variables DESC\_DISTRITO y DESC\_BARRIO.
9. Almacénalo en caché. Consulta en el puerto 4040 (UI de Spark) de tu usuario local el estadode los rdds almacenados.
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".
11. Elimina el registro en caché.
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.
13. Repite la función anterior utilizando funciones de ventana. (over(Window.partitionBy.....)).
14. Mediante una función Pivot muestra una tabla (que va a ser una tabla de contingencia) quecontenga 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:
15. Utilizando este df crea 3 columnas del portentaje 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
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.
17. Haz el mismo guardado pero en formato parquet. Compara el peso del archivo con el resultado anterior.

Spark y hive

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 tablasde Hive y haciendo todas las consultas sobre estas tablas sin transformarlas en ningún momento en DataFrames ni DataSets.

Cp /etc/hive/conf/hive-site.xml /etc/spark/conf/

Spark-shell

Import org.apache.spark.sql.hive.HiveContext

Val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)

SqlContext.sql(“show databases”).show(false)

SqlContext.sql(“use datos\_padron”)