Práctica PADRON

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
Time taken: 1.627 seconds
```

• 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.

Creación de la tabla con los parámetros que nos dicen

```
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,
    >     FX_CARGA TIMESTAMP,
    >     FX_DATOS_INI DATE,
    >     FX_DATOS_FIN DATE
    > )
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > WITH SERDEPROPERTIES (
    >     "separatorChar" = "\;",
    >     "quoteChar" = "\"",
    >     "escapeChar" = "\\"
    > )
    > STORED AS TEXTFILE
    > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.058 seconds
```

Cargamos los datos desde el csv a la tabla

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/padron.csv' INTO TABLE padron_txt;
Loading data to table datos_padron.padron_txt
Table datos_padron.padron_txt stats: [numFiles=1, totalSize=34057160]
OK
Time taken: 0.485 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.)

Create Table As Select (CTAS)

```
hive> CREATE TABLE padron_txt_2 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,
    >     FX_CARGA,
    >     FX_DATOS_INI,
    >     FX_DATOS_FIN
    > FROM padron_txt;
Query ID = cloudera_20240730115757_fd3ba867-d836-45eb-9e32-43d03be635f4
Total jobs = 3
Stage-Stage-1: Map: 1   Cumulative CPU: 4.31 sec   HDFS Read: 34062895 HDFS Write: 26422298 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 310 msec
OK
cod_distrito    desc_distrito   cod_dist_barrio desc_barrio     cod_barrio      cod_dist_seccion        c
rga     fx_datos_ini    fx_datos_fin
Time taken: 19.981 seconds
```

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

La palabra clave `LOCAL` en el comando `LOAD DATA` en Hive tiene una importancia significativa en cuanto a la ubicación de los datos que estás cargando en la tabla de Hive.

- El comando `LOAD DATA` en Hive se utiliza para mover datos desde un archivo en el sistema de archivos al directorio de la tabla en el sistema de archivos de Hadoop (HDFS).

```
LOAD DATA [LOCAL] INPATH 'path' [OVERWRITE] INTO TABLE table_name;
```

- Cuando usas `LOAD DATA` sin la palabra clave `LOCAL`, Hive asume que el archivo o directorio especificado está ubicado en el sistema de archivos distribuido de Hadoop (HDFS). En este caso, el archivo debe estar en HDFS, no en el sistema de archivos local del nodo donde se ejecuta Hive.

```
LOAD DATA INPATH '/user/hive/warehouse/data.csv' INTO TABLE padron_txt;
```

• 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.

```
hive> SELECT
    >     COUNT(*) AS total_rows,
    >     COUNT(CASE WHEN TRIM(ESPANOLESHOMBRES) = '' THEN 1 END) AS empty_esp_hombres,
    >     COUNT(CASE WHEN TRIM(ESPANOLESMUJERES) = '' THEN 1 END) AS empty_esp_mujeres,
    >     COUNT(CASE WHEN TRIM(EXTRANJEROSHOMBRES) = '' THEN 1 END) AS empty_extranjero_hombres,
    >     COUNT(CASE WHEN TRIM(EXTRANJEROSMUJERES) = '' THEN 1 END) AS empty_extranjero_mujeres
    > FROM padron_txt;
OK
total_rows      empty_esp_hombres       empty_esp_mujeres       empty_extranjero_hombres        empty_extranjero_mujeres
239342  0       0       0       0
Time taken: 26.265 seconds, Fetched: 1 row(s)
```

Como se puede ver, por la manera en la que se han tratado, no hay espacios en blanco.

• 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.

```
hive> CREATE TABLE padron_txt_2 (
    >     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 STRING,
    >     ESPANOLESHOMBRES INT,
    >     ESPANOLESMUJERES INT,
    >     EXTRANJEROSHOMBRES INT,
    >     EXTRANJEROSMUJERES INT,
    >     FX_CARGA TIMESTAMP,
    >     FX_DATOS_INI DATE,
    >     FX_DATOS_FIN DATE
    > )
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
    > WITH SERDEPROPERTIES (
    >     "input.regex" = "^(\\d+)\\;\"([^\"]*)\"\\;(\\d+)\\;\"([^\"]*)\"\\;(\\d+)\\;(\\d+)\\;(\\d+)\\;\"([^\"]*)\"\\;(\\d+)\\;(\\d+)\\;(\\d+)\\;(\\d+)\\;\
"([^\"]*)\"\\;\"([^\"]*)\"\\;\"([^\"]*)\"$"
    > )
    > STORED AS TEXTFILE;
```

La inserción de datos, los metemos omitiendo el header, par que no de NULL los campos del header, y luego lo volvemos a añadir los campos manual:

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/padron.csv' INTO TABLE padron_txt_2;
Loading data to table default.padron_txt_2
Table default.padron_txt_2 stats: [numFiles=1, totalSize=34056941]
OK
Time taken: 0.475 seconds
```

```
padron_txt_2.cod_distrito       padron_txt_2.desc_distrito      padron_txt_2.cod_dist_barrio    padron_txt_2.desc_barrio        padron_txt_2.cod_barrio pad
ron_txt_2.cod_dist_seccion      padron_txt_2.cod_seccion        padron_txt_2.cod_edad_int       padron_txt_2.espanoleshombres   padron_txt_2.espanolesmujer
es      padron_txt_2.extranjeroshombres padron_txt_2.extranjerosmujeres padron_txt_2.fx_carga   padron_txt_2.fx_datos_ini       padron_txt_2.fx_datos_fin
2       ARGANZUELA              202     ACACIAS                 2       2094    94              47      6       16      0       0       2024-07-07 00:00:01
.8166596        2024-07-01      2024-07-01
18      VILLA DE VALLECAS       1802    SANTA EUGENIA           2       18029   29              77      7       3       0       2       2024-07-07 00:00:01
.8166596        2024-07-01      2024-07-01
2       ARGANZUELA              205     DELICIAS                5       2062    62              71      6       6       0       2       2024-07-07 00:00:01
.8166596        2024-07-01      2024-07-01
9       MONCLOA-ARAVACA         907     ARAVACA                 7       9083    83              55      15      14      0       0       2024-07-07 00:00:01
.8166596        2024-07-01      2024-07-01
3       RETIRO                  306     NIÑO JESUS              6       3080    80              60      7       8       1       1       2024-07-07 00:00:01
.8166596        2024-07-01      2024-07-01
Time taken: 0.066 seconds, Fetched: 5 row(s)
```

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 y padron2 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.

A priori y por lo que he comprobado, lo tengo listo para trabajar

2- Investigamos el formato columnar parquet.

• 2.1) ¿Qué es CTAS?

CTAS significa "Create Table As Select" y es una técnica en SQL (y otras bases de datos) utilizada para crear una nueva tabla basada en los resultados de una consulta SELECT. Este método es muy útil para copiar datos de una tabla a otra o para crear una nueva tabla con una estructura y datos específicos sin tener que definir manualmente cada columna y tipo de datos.

• 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
    >     COD_DISTRITO,
    >     DESC_DISTRITO,
    >     COD_DIST_BARRIO,
    >     DESC_BARRIO,
    >     COD_BARRIO,
    >     COD_DIST_SECCION,
    >     COD_SECCION,
    >     COD_EDAD_INT,
    >     ESPANOLESHOMBRES,
    >     ESPANOLESMUJERES,
    >     EXTRANJEROSHOMBRES,
    >     EXTRANJEROSMUJERES,
    >     FX_CARGA,
    >     FX_DATOS_INI,
    >     FX_DATOS_FIN
    >     FROM padron_txt;
Query ID = cloudera_20240731091414_728a4d71-9bdf-4efa-9361-59e51971002b
Total jobs = 3
```

Comprobamos que la creación ha ido bien:

```
hive> select * from padron_parquet limit 3;
OK
padron_parquet.cod_distrito     padron_parquet.desc_distrito    padron_parquet.cod_dist_barrio  padron_parquet.desc_barrio      padron_parquet.cod_barrio p
adron_parquet.cod_dist_seccion  padron_parquet.cod_seccion      padron_parquet.cod_edad_int     padron_parquet.espanoleshombres padron_parquet.espanolesmuj
eres    padron_parquet.extranjeroshombres       padron_parquet.extranjerosmujeres       padron_parquet.fx_carga padron_parquet.fx_datos_ini     padron_parq
uet.fx_datos_fin
NULL    DESC_DISTRITO   NULL    DESC_BARRIO     NULL    NULL    NULL    COD_EDAD_INT    NULL    NULL    NULL    NULL    NULL    FX_DATOS_INI    FX_DATOS_FI
N
2       "ARGANZUELA          "  202     "ACACIAS            "  2       2094    94      "        47"    6       16      0       0       NULL    "2024-07-01
"       "2024-07-01"
18      "VILLA DE VALLECAS   "  1802    "SANTA EUGENIA      "  2       18029   29      "        77"    7       3       0       2       NULL    "2024-07-01
"       "2024-07-01"
Time taken: 0.039 seconds, Fetched: 3 row(s)
```

• 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).

Parquet en Hive no soporta “date”, por lo que hay que hacer algún cambio:
En vez de date, lo metemos como timestamp el dato.

```
hive> CREATE TABLE padron_parquet_2 (
    >     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,
    >     FX_CARGA TIMESTAMP,
    >     FX_DATOS_INI STRING,
    >     FX_DATOS_FIN STRING
    > )
    > STORED AS PARQUET;
OK
Time taken: 0.087 seconds
hive>
    > INSERT INTO padron_parquet_2
    > 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,
    >     CASE WHEN TRIM(COD_EDAD_INT) = '' THEN 0 ELSE CAST(TRIM(COD_EDAD_INT) AS INT) END AS COD_EDAD_INT,
    >     ESPANOLESHOMBRES,
    >     ESPANOLESMUJERES,
    >     EXTRANJEROSHOMBRES,
    >     EXTRANJEROSMUJERES,
    >     FX_CARGA,
    >     FX_DATOS_INI,
    >     FX_DATOS_FIN
    > FROM padron_txt_2;
```

Comprobamos que ha ido bien la creación:

```
hive> select * from padron_parquet_2 limit 3;
OK
padron_parquet_2.cod_distrito   padron_parquet_2.desc_distrito  padron_parquet_2.cod_dist_barrio        padron_parquet_2.desc_barrio    padron_parquet_2.co
d_barrio        padron_parquet_2.cod_dist_seccion       padron_parquet_2.cod_seccion    padron_parquet_2.cod_edad_int   padron_parquet_2.espanoleshombres p
adron_parquet_2.espanolesmujeres        padron_parquet_2.extranjeroshombres     padron_parquet_2.extranjerosmujeres     padron_parquet_2.fx_carga       pad
ron_parquet_2.fx_datos_ini      padron_parquet_2.fx_datos_fin
2       ARGANZUELA      202     ACACIAS 2       2094    94      47      6       16      0       0       2024-07-07 00:00:01.8166596     2024-07-01      202
4-07-01
18      VILLA DE VALLECAS       1802    SANTA EUGENIA   2       18029   29      77      7       3       0       2       2024-07-07 00:00:01.8166596     202
4-07-01 2024-07-01
2       ARGANZUELA      205     DELICIAS        5       2062    62      71      6       6       0       2       2024-07-07 00:00:01.8166596     2024-07-012
024-07-01
Time taken: 0.074 seconds, Fetched: 3 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.

La tabla “mala”

```
hive> CREATE TABLE padron_parquet (
    >     COD_DISTRITO STRING,
    >     DESC_DISTRITO STRING,
    >     COD_DIST_BARRIO STRING,
    >     DESC_BARRIO STRING,
    >     COD_BARRIO STRING,
    >     COD_DIST_SECCION STRING,
    >     COD_SECCION STRING,
    >     COD_EDAD_INT STRING,
    >     ESPANOLESHOMBRES STRING,
    >     ESPANOLESMUJERES STRING,
    >     EXTRANJEROSHOMBRES STRING,
    >     EXTRANJEROSMUJERES STRING,
    >     FX_CARGA STRING,
    >     FX_DATOS_INI STRING,
    >     FX_DATOS_FIN STRING
    > )
    > STORED AS PARQUET;
OK
Time taken: 0.085 seconds
```

```
hive> INSERT INTO padron_parquet SELECT * FROM padron_txt;
```

Comprobación:

```
hive> select * from padron_parquet limit 3;
OK
padron_parquet.cod_distrito     padron_parquet.desc_distrito    padron_parquet.cod_dist_barrio  padron_parquet.desc_barrio      padron_parquet.cod_barrio       padron_parquet.cod_dist_seccion padron_parquet.cod_sec
t.cod_edad_int  padron_parquet.espanoleshombres padron_parquet.espanolesmujeres padron_parquet.extranjeroshombres       padron_parquet.extranjerosmujeres       padron_parquet.fx_carga padron_parquet.fx_datos_ini
os_fin
NULL    DESC_DISTRITO   NULL    DESC_BARRIO     NULL    NULL    NULL    COD_EDAD_INT    NULL    NULL    NULL    NULL    NULL    FX_DATOS_INI    FX_DATOS_FIN
2       "ARGANZUELA             "  202     "ACACIAS                "  2       2094    94      "       47"     6       16      0       0       NULL    "2024-07-01"    "2024-07-01"
18      "VILLA DE VALLECAS      "  1802    "SANTA EUGENIA          "  2       18029   29      "       77"     7       3       0       2       NULL    "2024-07-01"    "2024-07-01"
Time taken: 0.047 seconds, Fetched: 3 row(s)
```

Ahora la tabla “buena”

```
hive> CREATE TABLE padron_parquet_2 (
    >     COD_DISTRITO STRING,
    >     DESC_DISTRITO STRING,
    >     COD_DIST_BARRIO STRING,
    >     DESC_BARRIO STRING,
    >     COD_BARRIO STRING,
    >     COD_DIST_SECCION STRING,
    >     COD_SECCION STRING,
    >     COD_EDAD_INT STRING,
    >     ESPANOLESHOMBRES STRING,
    >     ESPANOLESMUJERES STRING,
    >     EXTRANJEROSHOMBRES STRING,
    >     EXTRANJEROSMUJERES STRING,
    >     FX_CARGA STRING,
    >     FX_DATOS_INI STRING,
    >     FX_DATOS_FIN STRING
    > )
    > STORED AS PARQUET;
OK
Time taken: 0.049 seconds
```

```
hive> INSERT INTO padron_parquet_2 SELECT * FROM padron_txt_2;
```

Comprobación:

```
hive> select * from padron_parquet_2 limit 3;
OK
padron_parquet_2.cod_distrito   padron_parquet_2.desc_distrito  padron_parquet_2.cod_dist_barrio        padron_parquet_2.desc_barrio    padron_parquet_2.cod_barrio     padron_parquet_2.cod_dist_seccion       padron
adron_parquet_2.cod_edad_int    padron_parquet_2.espanoleshombres       padron_parquet_2.espanolesmujeres       padron_parquet_2.extranjeroshombres     padron_parquet_2.extranjerosmujeres     padron_parquet_2.fx_ca
t_2.fx_datos_ini        padron_parquet_2.fx_datos_fin
2       ARGANZUELA              202     ACACIAS                 2       2094    94              47      6       16      0       0       2024-07-07 00:00:01.8166596     2024-07-01      2024-07-01
18      VILLA DE VALLECAS       1802    SANTA EUGENIA           2       18029   29              77      7       3       0       2       2024-07-07 00:00:01.8166596     2024-07-01      2024-07-01
2       ARGANZUELA              205     DELICIAS                5       2062    62              71      6       6       0       2       2024-07-07 00:00:01.8166596     2024-07-01      2024-07-01
Time taken: 0.047 seconds, Fetched: 3 row(s)
```

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

Parquet es un formato de almacenamiento columnar que organiza los datos por columnas en lugar de filas. Esto permite una compresión eficiente y reduce el tamaño del almacenamiento. Su diseño facilita la lectura rápida de solo las columnas necesarias para consultas, mejorando el rendimiento en el procesamiento de datos. Además, Parquet es compatible con herramientas de Big Data como Apache Spark y Hive, y permite la evolución del esquema sin afectar los datos existentes.

• 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").

Para ello se pillará de cada tabla su LOCATION, accediendo a ellas con el siguente comando:

```
hive> SHOW CREATE TABLE PADRON_TXT;
```

```
LOCATION
  'hdfs://quickstart.cloudera:8020/user/hive/warehouse/padron_txt'
```

```
LOCATION
  'hdfs://quickstart.cloudera:8020/user/hive/warehouse/padron_txt_2'
```

```
LOCATION
  'hdfs://quickstart.cloudera:8020/user/hive/warehouse/padron_parquet'
```

```
LOCATION
  'hdfs://quickstart.cloudera:8020/user/hive/warehouse/padron_parquet_2'
```

```
[cloudera@quickstart ~]$ hdfs dfs -du -h /user/hive/warehouse/padron_txt;
32.5 M  32.5 M  /user/hive/warehouse/padron_txt/padron.csv
[cloudera@quickstart ~]$ hdfs dfs -du -h /user/hive/warehouse/padron_txt_2;
32.5 M  32.5 M  /user/hive/warehouse/padron_txt_2/padron.csv
[cloudera@quickstart ~]$ hdfs dfs -du -h /user/hive/warehouse/padron_parquet;
2.4 M  2.4 M  /user/hive/warehouse/padron_parquet/000000_0
[cloudera@quickstart ~]$ hdfs dfs -du -h /user/hive/warehouse/padron_parquet_2;
2.4 M  2.4 M  /user/hive/warehouse/padron_parquet_2/000000_0
```

Se puede ver que Parquet ocupa 30 M menos que txt, lo que es sensacional para trabajar con datos grandes.

3- Juguemos con Impala.

• 3.1) ¿Qué es Impala?

Impala es un motor de consultas SQL de código abierto para Hadoop, diseñado para ofrecer consultas rápidas y analíticas en grandes volúmenes de datos. Lee datos directamente desde HDFS y Apache HBase, usa procesamiento en memoria para mejorar la velocidad, y es compatible con SQL estándar.

Es ideal para análisis en tiempo real, escalabilidad horizontal y se integra con herramientas de BI para crear dashboards y reportes.

• 3.2) ¿En qué se diferencia de Hive?

Impala y Hive son motores SQL para Hadoop con diferencias notables. Impala es más rápido y ofrece consultas interactivas en tiempo real al procesar datos en memoria, mientras que Hive, que usa MapReduce o Tez, puede ser más lento debido a la escritura en disco. Impala también se integra bien con HBase, permitiendo consultas sobre datos NoSQL, mientras que Hive se usa principalmente con HDFS. Impala está optimizado para consultas rápidas y complejas, mientras que Hive es más adecuado para procesamiento por lotes y tareas ETL.

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

Se utiliza para forzar la actualización de la caché de metadatos de la base de datos. Esto es necesario cuando los metadatos de las tablas o particiones han cambiado fuera de Impala, como al agregar, eliminar o modificar archivos de datos. Ejecutar este comando garantiza que Impala lea los metadatos actualizados y refleje los cambios en las consultas posteriores.

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

Entro en Impala:

```
[cloudera@quickstart ~]$ impala-shell
Starting Impala Shell without Kerberos authentication
Connected to quickstart.cloudera:21000
Server version: impalad version 2.10.0-cdh5.13.0 RELEASE (build 2511805f1eaa991df1460276c7e9f19d819cd4e4)
***********************************************************************************
Welcome to the Impala shell.
(Impala Shell v2.10.0-cdh5.13.0 (2511805) built on Wed Oct  4 10:55:37 PDT 2017)

Want to know what version of Impala you're connected to? Run the VERSION command to
find out!
***********************************************************************************
```

```
[quickstart.cloudera:21000] > INVALIDATE METADATA datos_padron.padron_txt;
Query: invalidate METADATA datos_padron.padron_txt
Query submitted at: 2024-07-31 12:51:39 (Coordinator: http://quickstart.cloudera:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?query_id=e94f811570d74aff:87a7489c00000000
Fetched 0 row(s) in 0.17s
```

```
[quickstart.cloudera:21000] > INVALIDATE METADATA datos_padron.padron_txt_2;
Query: invalidate METADATA datos_padron.padron_txt_2
Query submitted at: 2024-07-31 12:52:03 (Coordinator: http://quickstart.cloudera:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?query_id=4245a405523548f1:5500b64b00000000
Fetched 0 row(s) in 0.12s
```

• 3.5) 3.6) Calcular el total de EspanolesHombres, espanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC_DISTRITO y DESC_BARRIO.

Llevar a cabo las consultas en Hive en las tablas padron_txt_2 y padron_parquet_2 (No deberían incluir espacios innecesarios). ¿Alguna conclusión?

```
hive> SELECT
    >     DESC_DISTRITO,
    >     DESC_BARRIO,
    >     SUM(ESPANOLESHOMBRES) AS Total_EspanolesHombres,
    >     SUM(ESPANOLESMUJERES) AS Total_EspanolesMujeres,
    >     SUM(EXTRANJEROSHOMBRES) AS Total_ExtranjerosHombres,
    >     SUM(EXTRANJEROSMUJERES) AS Total_ExtranjerosMujeres
    > FROM
    >     datos_padron.padron_txt_2
    > GROUP BY
    >     DESC_DISTRITO,
    >     DESC_BARRIO;
```

```
desc_distrito   desc_barrio     total_espanoleshombres  total_espanolesmujeres  total_extranjeroshombres        total_extranjerosmujeres
                2.0     4.0     1.0     0.0
ARGANZUELA                      0.0     0.0     1.0     0.0
ARGANZUELA              ACACIAS                 15297.0 17871.0 1636.0  1895.0
ARGANZUELA              ATOCHA                  899.0   921.0   129.0   152.0
ARGANZUELA              CHOPERA                 7786.0  9357.0  1635.0  1806.0
```

Ahora en parquet…

```
hive> SELECT
    >     DESC_DISTRITO,
    >     DESC_BARRIO,
    >     SUM(ESPANOLESHOMBRES) AS Total_EspanolesHombres,
    >     SUM(ESPANOLESMUJERES) AS Total_EspanolesMujeres,
    >     SUM(EXTRANJEROSHOMBRES) AS Total_ExtranjerosHombres,
    >     SUM(EXTRANJEROSMUJERES) AS Total_ExtranjerosMujeres
    > FROM
    >     padron_parquet_2
    > GROUP BY
    >     DESC_DISTRITO,
    >     DESC_BARRIO;
```

```
desc_distrito   desc_barrio     total_espanoleshombres  total_espanolesmujeres  total_extranjeroshombres        total_extranjerosmujeres
NULL    NULL    NULL    NULL    NULL    NULL
ARGANZUELA              ACACIAS                 15297.0 17871.0 1636.0  1895.0
ARGANZUELA              ATOCHA                  899.0   921.0   129.0   152.0
ARGANZUELA              CHOPERA                 7786.0  9357.0  1635.0  1806.0
ARGANZUELA              DELICIAS                11928.0 13417.0 2098.0  2331.0
```

• 3.7) 3.8) Llevar a cabo la misma consulta sobre las mismas tablas en Impala. ¿Alguna conclusión?

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

```
[quickstart.cloudera:21000] > SELECT
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO,
                            >     SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_Es
panolesHombres,
                            >     SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_Es
panolesMujeres,
                            >     SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_
ExtranjerosHombres,
                            >     SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_
ExtranjerosMujeres
                            > FROM
                            >     default.padron_parquet_2
                            > GROUP BY
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO;
```

```
+---------------------+---------------------+-----------------------+--------
---------------+-------------------------+-------------------------+
| desc_distrito       | desc_barrio         | total_espanoleshombres | total_e
spanolesmujeres | total_extranjeroshombres | total_extranjerosmujeres |
+---------------------+---------------------+-----------------------+--------
---------------+-------------------------+-------------------------+
| ARGANZUELA          | LEGAZPI             | 8956                   | 9338
               | 859                      | 950                     |
| CHAMBERI            | RIOS ROSAS          | 10525                  | 13407
               | 1915                     | 2397                    |
| BARAJAS             | CASCO H.BARAJAS     | 2983                   | 3182
               | 832                      | 955                     |
| USERA               | ORCASITAS           | 9758                   | 10971
               | 1480                     | 1650                    |
| BARAJAS             | TIMON               | 5276                   | 5805
```

En impala he percibido que la ejecución se ha realizado mucho más rápido que en hive, me ha sorprendido lo rápido que ha extraído los datos. Impala es más rápido que Hive porque ejecuta consultas en memoria y usa un motor de procesamiento distribuido optimizado, mientras que Hive utiliza MapReduce, que es más lento debido a su enfoque en disco.

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 STRING,
    >     ESPANOLESHOMBRES INT,
    >     ESPANOLESMUJERES INT,
    >     EXTRANJEROSHOMBRES INT,
    >     EXTRANJEROSMUJERES INT,
    >     FX_CARGA STRING,
    >     FX_DATOS_INI STRING,
    >     FX_DATOS_FIN STRING
    > )
    > PARTITIONED BY (DESC_DISTRITO STRING, DESC_BARRIO STRING)
    > STORED AS PARQUET;
OK
```

• 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.

Me ha dado un error en la inserción de datos ya que me indicaba que Hive intentaba crear demasiadas particiones dinámicas. Este problema es común cuando se intenta particionar por columnas con una alta cardinalidad sin ajustar los límites de particiones dinámicas.

Para resolverlo, he incrementado los límites de particion:

```
hive> SET hive.exec.max.dynamic.partitions = 1000;
hive> SET hive.exec.max.dynamic.partitions.pernode = 1000;
```

```
hive> INSERT INTO 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,
    >     FX_CARGA,
    >     FX_DATOS_INI,
    >     FX_DATOS_FIN,
    >     DESC_DISTRITO,
    >     DESC_BARRIO
    > FROM padron_parquet_2;
```

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

```
[quickstart.cloudera:21000] > INVALIDATE METADATA default.padron_particionado;
```

```
[quickstart.cloudera:21000] > SHOW TABLES;
Query: show TABLES
+----------------------+
| name                 |
+----------------------+
| empleados            |
| padron2              |
| padron_parquet       |
| padron_parquet_2     |
| padron_particionado  |
```

Vemos que aparece aqui.

• 4.4) • 4.5) 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.
Llevar a cabo la consulta en Hive en las tablas padron_parquet y padron_partitionado. ¿Alguna conclusión?

```
hive> SELECT
    >     DESC_DISTRITO,
    >     DESC_BARRIO,
    >     SUM(ESPANOLESHOMBRES) AS Total_EspanolesHombres,
    >     SUM(ESPANOLESMUJERES) AS Total_EspanolesMujeres,
    >     SUM(EXTRANJEROSHOMBRES) AS Total_ExtranjerosHombres,
    >     SUM(EXTRANJEROSMUJERES) AS Total_ExtranjerosMujeres
    > FROM
    >     default.padron_parquet
    > WHERE
    >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
    > GROUP BY
    >     DESC_DISTRITO,
    >     DESC_BARRIO;
```

```
hive> SELECT
    >     DESC_DISTRITO,
    >     DESC_BARRIO,
    >     SUM(ESPANOLESHOMBRES) AS Total_EspanolesHombres,
    >     SUM(ESPANOLESMUJERES) AS Total_EspanolesMujeres,
    >     SUM(EXTRANJEROSHOMBRES) AS Total_ExtranjerosHombres,
    >     SUM(EXTRANJEROSMUJERES) AS Total_ExtranjerosMujeres
    > FROM
    >     default.padron_particionado
    > WHERE
    >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
    > GROUP BY
    >     DESC_DISTRITO,
    >     DESC_BARRIO;
```

La particionada se realiza a mayor velocidad, ya que al estar particonada, no tiene que leer toda la tabla completa, por lo que es más eficiente, por lo que la tabla particionada, mejora el rendimiento de la consulta.

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

```
[quickstart.cloudera:21000] > SELECT
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO,
                            >     SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
                            >     SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
                            >     SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
                            >     SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
                            > FROM
                            >     default.padron_parquet
                            > WHERE
                            >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
                            > GROUP BY
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO;
Query: select DESC_DISTRITO,
    DESC_BARRIO,
    SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
    SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
    SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
    SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
FROM
    default.padron_parquet
WHERE
    DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
GROUP BY
    DESC_DISTRITO,
    DESC_BARRIO
Query submitted at: 2024-07-31 14:11:32 (Coordinator: http://quickstart.cloudera:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?query_id=e645ae6b285864d8:9ce80cf100000000
Fetched 0 row(s) in 0.39s
```

```
[quickstart.cloudera:21000] > SELECT
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO,
                            >     SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
                            >     SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
                            >     SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
                            >     SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
                            > FROM
                            >     default.padron_particionado
                            > WHERE
                            >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
                            > GROUP BY
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO;
Query: select DESC_DISTRITO,
    DESC_BARRIO,
    SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
    SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
    SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
    SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
FROM
    default.padron_particionado
WHERE
    DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
GROUP BY
    DESC_DISTRITO,
    DESC_BARRIO
Query submitted at: 2024-07-31 14:15:09 (Coordinator: http://quickstart.cloudera:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?query_id=4e4072b79fd0423e:107ba00d00000000
Fetched 0 row(s) in 4.45s
```

En Impala, la ejecución de consultas es mucho más rápida que en Hive, ya que Impala procesa datos en memoria con un motor distribuido optimizado, mientras que Hive usa MapReduce, más lento por su enfoque en disco. Además, las tablas particionadas en Hive mejoran el rendimiento porque solo leen las particiones relevantes en lugar de toda la tabla.

• 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.

En impala:

```
[quickstart.cloudera:21000] > SELECT
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO,
                            >     SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
                            >     SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
                            >     SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
                            >     SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
                            > FROM
                            >     default.padron_parquet_2
                            > WHERE
                            >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
                            > GROUP BY
                            >     DESC_DISTRITO,
                            >     DESC_BARRIO;
Query: select DESC_DISTRITO,
    DESC_BARRIO,
    SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
    SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
    SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
    SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
FROM
    default.padron_parquet_2
WHERE
    DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
GROUP BY
    DESC_DISTRITO,
    DESC_BARRIO
Query submitted at: 2024-07-31 14:19:10 (Coordinator: http://quickstart.cloudera:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?query_id=524a4648130af07a:dde989500000000
Fetched 0 row(s) in 0.43s
```

En hive:

```
hive> SELECT
    >     DESC_DISTRITO,
    >     DESC_BARRIO,
    >     SUM(CAST(ESPANOLESHOMBRES AS INT)) AS Total_EspanolesHombres,
    >     SUM(CAST(ESPANOLESMUJERES AS INT)) AS Total_EspanolesMujeres,
    >     SUM(CAST(EXTRANJEROSHOMBRES AS INT)) AS Total_ExtranjerosHombres,
    >     SUM(CAST(EXTRANJEROSMUJERES AS INT)) AS Total_ExtranjerosMujeres
    > FROM
    >     default.padron_txt_2
    > WHERE
    >     DESC_DISTRITO IN ('CENTRO', 'LATINA', 'CHAMARTIN', 'TETUAN', 'VICALVARO', 'BARAJAS')
    > GROUP BY
    >     DESC_DISTRITO,
    >     DESC_BARRIO;
```

En memoria con un motor distribuido optimizado, mientras que Hive usa MapReduce, más lento por su enfoque en disco. Además, las tablas particionadas en Hive mejoran el rendimiento porque solo leen las particiones relevantes en lugar de toda la tabla.

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

datos1.txt

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

datos2.txt

• 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 ~]$ hdfs dfs -mkdir /user/cloudera/test
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
Found 5 items
drwxrwxrwx   - cloudera cloudera          0 2024-07-10 11:42 /user/cloudera/ejercicios
drwxrwxrwx   - cloudera cloudera          0 2024-07-11 14:05 /user/cloudera/ejercicios_master
drwxrwxrwx   - cloudera cloudera          0 2024-07-10 13:37 /user/cloudera/terasort-input
drwxrwxrwx   - cloudera cloudera          0 2024-07-10 13:38 /user/cloudera/terasort-output
drwxr-xr-x   - cloudera cloudera          0 2024-07-31 14:31 /user/cloudera/test
```

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

```
[cloudera@quickstart ~]$ cd /home/cloudera/Desktop/Padron/
[cloudera@quickstart Padron]$ hdfs dfs -put datos1.txt /user/cloudera/test/
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test/
Found 1 items
-rw-r--r--   1 cloudera cloudera         18 2024-07-31 14:33 /user/cloudera/test/datos1.txt
```

• 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: 0.343 seconds
hive> USE numeros;
OK
Time taken: 0.023 seconds
hive> CREATE TABLE numeros_tbl (
    >     num1 INT,
    >     num2 INT,
    >     num3 INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n';
OK
Time taken: 0.244 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?

Inserción de datos…

```
hive> LOAD DATA INPATH '/user/cloudera/test/datos1.txt' INTO TABLE numeros_tbl;
Loading data to table numeros.numeros_tbl
Table numeros.numeros_tbl stats: [numFiles=1, totalSize=18]
OK
Time taken: 0.311 seconds
hive> select * from numeros_tbl;
OK
numeros_tbl.num1        numeros_tbl.num2        numeros_tbl.num3
1       2       3
4       5       6
7       8       9
Time taken: 0.285 seconds, Fetched: 3 row(s)
```

Vemos donde estaban almacenados los datos antes…

```
hive> DESCRIBE FORMATTED numeros_tbl;
```

```
Location:               hdfs://quickstart.cloudera:8020/user/hive/warehouse/numeros.db/numeros_tbl
```

Vemos que después de eliminar la tabla no aparece:

```
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test/;
```

Esto se debe a…

1. La tabla en Hive no es externa: En Hive, si la tabla no está definida como externa y eliminas la tabla con `DROP TABLE`, Hive eliminará automáticamente los datos almacenados en HDFS asociados a esa tabla.
2. Ubicación predeterminada de la tabla: Si no se especifica una ubicación externa personalizada para la tabla, los datos se almacenan en la ubicación predeterminada de Hive, que se encuentra en `/user/hive/warehouse` (o similar). Al eliminar la tabla, Hive elimina tanto la definición de la tabla como los datos en la ubicación predeterminada. **(es el caso)**

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

```
[cloudera@quickstart Padron]$ hdfs dfs -put datos1.txt /user/cloudera/test
```

```
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test
Found 1 items
-rw-r--r--   1 cloudera cloudera         18 2024-07-31 14:48 /user/cloudera/test
/datos1.txt
```

• 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_externos (
    >     col1 INT,
    >     col2 INT,
    >     col3 INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n';
OK
Time taken: 1.52 seconds
```

```
hive> LOAD DATA INPATH '/user/cloudera/test/datos1.txt' INTO TABLE numeros_exter
nos;
Loading data to table numeros.numeros_externos
Table numeros.numeros_externos stats: [numFiles=1, numRows=0, totalSize=18, rawD
ataSize=0]
OK
Time taken: 0.225 seconds
```

Una vez hecha la inserción, se puede comprobar que los datos están en hdfs:

```
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test
Found 1 items
-rwxr-xr-x   1 cloudera cloudera         18 2024-08-01 08:18 /user/cloudera/test
/datos1.txt
```

Borro la tabla…

```
hive> drop table numeros_externos;
OK
Time taken: 0.359 seconds
```

Compruebo si siguen los datos…

```
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test
Found 1 items
-rwxr-xr-x   1 cloudera cloudera         18 2024-08-01 08:18 /user/cloudera/test
/datos1.txt
```

Los datos permanecen en HDFS a pesar de eliminar la tabla externa en Hive porque:

- **Tabla externa**: Al crear una tabla externa, Hive solo gestiona la metadata y no los datos.
- **Eliminar la tabla**: Eliminar una tabla externa en Hive solo elimina la metadata de la tabla, pero no los datos en HDFS.

Así, los datos siguen existiendo en HDFS después de la eliminación de la tabla externa en 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?

```
[cloudera@quickstart Padron]$ hdfs dfs -rm /user/cloudera/test/datos1.txt
Deleted /user/cloudera/test/datos1.txt
[cloudera@quickstart Padron]$ hdfs dfs -put datos1.txt /user/cloudera/test/
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test
Found 1 items
-rw-r--r--   1 cloudera cloudera         18 2024-08-01 08:26 /user/cloudera/test
/datos1.txt
```

```
hive> CREATE EXTERNAL TABLE numeros_externos (
    >     num1 INT,
    >     num2 INT,
    >     num3 INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > LOCATION '/user/cloudera/test';
OK
Time taken: 0.066 seconds
hive> SELECT * FROM numeros_externos;
OK
numeros_externos.num1   numeros_externos.num2   numeros_externos.num3
1       2       3
4       5       6
7       8       9
Time taken: 0.37 seconds, Fetched: 3 row(s)
```

La tabla externa se refiere directamente a los datos en HDFS. No es necesario cargar los datos explícitamente en la tabla ya que Hive mapea directamente al archivo o directorio especificado en `LOCATION`. Por lo tanto, al hacer la consulta, los datos presentes en el archivo `datos1` en el directorio `/test` serán mostrados.

• 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 Padron]$ hdfs dfs -put datos2.txt /user/cloudera/test/
[cloudera@quickstart Padron]$ hdfs dfs -ls /user/cloudera/test
Found 2 items
-rw-r--r--   1 cloudera cloudera         18 2024-08-01 08:26 /user/cloudera/test
/datos1.txt
-rw-r--r--   1 cloudera cloudera         27 2024-08-01 08:34 /user/cloudera/test
/datos2.txt
```

```
hive> SELECT * FROM numeros_externos;
OK
numeros_externos.num1   numeros_externos.num2   numeros_externos.num3
1       2       3
4       5       6
7       8       9
10      21      32
43      54      65
76      87      98
Time taken: 0.06 seconds, Fetched: 6 row(s)
```

Como insertamos de nuevo en la carpeta a la que hace referencia la tabla, todos los datos que metamos dentro de esa carpeta, los podremos utilizar ya que ta tabla pilla su referencia, por lo que podemos sacar como conclusión que trabajar con 'LOCATION', es una forma bastante eficiente y fácil de usar, que nos permite trabajar de una manera más cómoda.

• 5.11) Extrae conclusiones de todos estos anteriores apartados.

1. Rendimiento de Hive vs. Impala:
    - Impala: Observamos que las consultas en Impala se ejecutan mucho más rápido que en Hive. Esto se debe a que Impala ejecuta las consultas en memoria y utiliza un motor de procesamiento distribuido optimizado. Esto contrasta con Hive, que utiliza MapReduce, un enfoque más lento debido a su operación basada en disco.
    - Tablas particionadas: Las tablas particionadas ofrecen un mejor rendimiento en las consultas porque no necesitan leer toda la tabla, sino solo las particiones relevantes. Esto mejora significativamente la eficiencia de las consultas.
2. Operaciones con tablas en Hive:
    - Carga de datos en tablas Hive:
        - Al cargar datos en una tabla interna de Hive, los datos se mueven desde su ubicación original en HDFS al directorio gestionado por Hive. Esto hace que los datos desaparezcan de su ubicación original una vez que se cargan en la tabla.
        - Si se elimina una tabla interna, los datos almacenados en HDFS también se eliminan.
    - Tablas externas en Hive:
        - Al crear y cargar datos en una tabla externa, Hive no mueve los datos, sino que los referencia directamente desde su ubicación original en HDFS. Esto significa que los datos permanecen en su lugar original.
        - Al eliminar una tabla externa, los datos en HDFS no se eliminan porque Hive no es el propietario de los datos, solo los referencia.
3. Gestión de archivos en HDFS:
    - Borrar y mover archivos en HDFS:
        - Los comandos HDFS (`hdfs dfs -rm` y `hdfs dfs -put`) permiten gestionar los archivos y directorios en el sistema de archivos distribuido.
    - Creación de directorios y archivos:
        - Creamos directorios en HDFS para organizar y almacenar datos, facilitando su referencia desde Hive.
    - Persistencia de datos:
        - Los datos cargados en tablas externas en Hive permanecen en su ubicación original en HDFS incluso después de eliminar la tabla, lo cual es útil para la gestión y persistencia de grandes volúmenes de datos.
4. Resumen general:
    - Impala es altamente eficiente para consultas rápidas debido a su enfoque en memoria.
    - Hive es útil para operaciones ETL y manipulación de datos debido a su integración con Hadoop y su capacidad para manejar grandes volúmenes de datos.
    - Tablas particionadas mejoran el rendimiento al reducir la cantidad de datos leídos durante las consultas.
    - Tablas externas en Hive proporcionan flexibilidad para referenciar datos en HDFS sin moverlos, asegurando que los datos permanezcan accesibles incluso si la tabla se elimina.

Estas conclusiones resaltan las ventajas y consideraciones prácticas al utilizar herramientas de Big Data como Hive e Impala, así como la importancia de una adecuada gestión de datos en HDFS para un rendimiento y eficiencia óptimos.

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- SPARK

• 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.

```
%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim, when

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

df = spark.read.csv("/FileStore/tables/Formacion_Binaia/Practica_Padron/estadisticas202407.csv",
                    header=True,   #primera linea de archivo son columnas
                    inferSchema=True, #permite a spark inferir el tipo de datos de cada columna
                    sep=';', #separacion de campos
                    quote='"',   #indica que las comillas se usan para envolver campos
                    ignoreLeadingWhiteSpace=True, #ignora espacios en blanco al principio
                    ignoreTrailingWhiteSpace=True,   #y al final de los campos
                    nullValue='',   #celdas vacias como valores nulos
                    nanValue='0')   #sustituye valores nan por 0

# Reemplazar valores nulos con 0
df = df.fillna(0)

# Eliminar espacios innecesarios en los campos
df = df.select([trim(col(c)).alias(c) for c in df.columns])

# Mostrar el esquema para confirmar que se infirió correctamente
df.printSchema()

# Mostrar algunas filas para verificar los datos
df.show(10)
```

```
+------------+------------------+----------------+--------------------+------------+------------+
|COD_DISTRITO|     DESC_DISTRITO|COD_DIST_BARRIO|  DESC_BARRIO|COD_BARRIO|COD_DIST_SECCION|COD_SECCION|COD_EDAD_INT|ESPANOLESHOMBRES|ESPAN
OLESMUJERES|EXTRANJEROSHOMBRES|EXTRANJEROSMUJERES|           FX_CARGA|FX_DATOS_INI|FX_DATOS_FIN|
+------------+------------------+----------------+--------------------+----------+----------------+-----------+------------+----------------+-----
-----------+------------------+------------------+--------------------+------------+------------+
|           2|        ARGANZUELA|             202|      ACACIAS|         2|            2094|         94|          47|               6|
16|                 0|                 0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|          18|  VILLA DE VALLECAS|            1802|SANTA EUGENIA|         2|           18029|         29|          77|               7|
3|                 0|                 2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|           2|        ARGANZUELA|             205|     DELICIAS|         5|            2062|         62|          71|               6|
6|                 0|                 2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|           9|    MONCLOA-ARAVACA|             907|      ARAVACA|         7|            9083|         83|          55|              15|
14|                 0|                 0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|           3|            RETIRO|             306|   NIÑO JESUS|         6|            3080|         80|          60|               7|
8|                 1|                 1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|           4|         SALAMANCA|             405|        LISTA|         5|            4102|        102|          78|               1|
6|                 0|                 0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
```

• 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.

```
%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim, regexp_replace

# Crear una sesión de Spark
spark = SparkSession.builder.appName("Padron").getOrCreate()

# Leer el archivo CSV en un DataFrame de Spark con menos opciones de tratamiento inicial
df = spark.read.csv("/FileStore/tables/Formacion_Binaia/Practica_Padron/estadisticas202407.csv", header=True, sep=';')

# Mostrar el esquema inicial para ver cómo se cargaron los datos
df.printSchema()

# Reemplazar comillas dobles y espacios innecesarios en los campos
df = df.select([trim(regexp_replace(col(c), '"', '')).alias(c) for c in df.columns])

# Reemplazar valores vacíos con 0
df = df.fillna('0')

# Convertir las columnas adecuadas a int
int_columns = ['ESPANOLESHOMBRES', 'ESPANOLESMUJERES', 'EXTRANJEROSHOMBRES', 'EXTRANJEROSMUJERES']
for column in int_columns:
    df = df.withColumn(column, col(column).cast('int'))

# Mostrar el esquema para confirmar que se infirió correctamente
df.printSchema()

# Mostrar algunas filas para verificar los datos
df.show(10)
```

```
+-----------+-----------------+--------------+------------+----------+---------------+-----------+-----------+---------------+-----
----------+-----------------+-----------------+--------------------+-----------+-----------+
|COD_DISTRITO|     DESC_DISTRITO|COD_DIST_BARRIO|  DESC_BARRIO|COD_BARRIO|COD_DIST_SECCION|COD_SECCION|COD_EDAD_INT|ESPANOLESHOMBRES|ESPAN
OLESMUJERES|EXTRANJEROSHOMBRES|EXTRANJEROSMUJERES|           FX_CARGA|FX_DATOS_INI|FX_DATOS_FIN|
+-----------+-----------------+--------------+------------+----------+---------------+-----------+-----------+---------------+-----
----------+-----------------+-----------------+--------------------+-----------+-----------+
|          2|        ARGANZUELA|           202|     ACACIAS|         2|           2094|         94|         47|              6|
16|                0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|         18|  VILLA DE VALLECAS|          1802|SANTA EUGENIA|         2|          18029|         29|         77|              7|
3|                0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|          2|        ARGANZUELA|           205|     DELICIAS|         5|           2062|         62|         71|              6|
6|                0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|          9|    MONCLOA-ARAVACA|           907|     ARAVACA|         7|           9083|         83|         55|             15|
14|                0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|
|          3|           RETIRO|           306|   NIÑO JESUS|         6|           3080|         80|         60|              7|
```

• 6.3) Enumera todos los barrios diferentes.

```
%python

# Enumerar todos los barrios diferentes
barrios_unicos = df.select('DESC_BARRIO').distinct().orderBy('DESC_BARRIO')

# Mostrar los barrios únicos
barrios_unicos.show(truncate=False)
```

```
▸ (2) Spark Jobs
▸ barrios_unicos: pyspark.sql.dataframe.DataFrame = [DESC_BARRIO: string]
|ABRANTES        |
|ACACIAS         |
|ADELFAS         |
|AEROPUERTO      |
|ALAMEDA DE OSUNA|
|ALMAGRO         |
|ALMENARA        |
|ALMENDRALES     |
|ALUCHE          |
|AMPOSTA         |
|ANGELES         |
|APOSTOL SANTIAGO|
|ARAPILES        |
|ARAVACA         |
|ARCOS           |
|ARGUELLES       |
|ATALAYA         |
|ATOCHA          |
|BELLAS VISTAS   |
+----------------+
only showing top 20 rows
```

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

```
%python
#Una vista temporal permite ejecutar consultas SQL sobre datos que ya están en un DataFrame de Spark, sin tener que realizar el procesamiento de datos nuevamente.
# Crear una vista temporal
df.createOrReplaceTempView("padron")

# Contar el número de barrios diferentes usando la vista temporal
count_barrios = spark.sql("""
    SELECT COUNT(DISTINCT DESC_BARRIO) AS num_barrios
    FROM padron
""")

# Mostrar el resultado
count_barrios.show()
```

(3) Spark Jobs

count_barrios: pyspark.sql.dataframe.DataFrame = [num_barrios: long]

```
+-----------+
|num_barrios|
+-----------+
|        131|
+-----------+
```

• 6.5) Crea una nueva columna que muestre la longitud de los campos de la columna DESC_DISTRITO y que se llame "longitud".

withcolumn, se usa para añadir una nueva columna o para reemplazar una columna existente con una nueva versión.

```
%python
from pyspark.sql.functions import length

# Crear una nueva columna 'longitud' que muestra la longitud de los valores en 'DESC_DISTRITO'
df = df.withColumn("longitud", length(col("DESC_DISTRITO")))

# Mostrar el esquema actualizado y algunas filas para verificar el resultado
df.printSchema()
df.show(10)
```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [COD_DISTRITO: string, DESC_DISTRITO: string ... 14 more fields]

```
+-----------+------------------+--------------+-------------+----------+----------------+-----------+------------+----------------+----------------+-
--+-----------------+--------------------+-----------+-----------+--------+
|          2|         ARGANZUELA|           202|      ACACIAS|         2|            2094|         94|          47|               6|              16|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|
|         18|  VILLA DE VALLECAS|          1802|SANTA EUGENIA|         2|           18029|         29|          77|               7|               3|
0|                2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      17|
|          2|         ARGANZUELA|           205|     DELICIAS|         5|            2062|         62|          71|               6|               6|
0|                2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|
|          9|    MONCLOA-ARAVACA|           907|      ARAVACA|         7|            9083|         83|          55|              15|              14|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      15|
|          3|             RETIRO|           306|   NIÑO JESUS|         6|            3080|         80|          60|               7|               8|
```

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

```
%python
from pyspark.sql.functions import lit

# Crear una nueva columna 'longitud' que muestra la longitud de los valores en 'DESC_DISTRITO'
df = df.withColumn("valor_fijo", lit(5))

# Mostrar el esquema actualizado y algunas filas para verificar el resultado
df.printSchema()
df.show(10)
```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [COD_DISTRITO: string, DESC_DISTRITO: string ... 15 more fields]

```
--+-----------------+--------------------+-----------+-----------+--------+---------+
|          2|         ARGANZUELA|           202|      ACACIAS|         2|            2094|         94|          47|               6|              16|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|        5|
|         18|  VILLA DE VALLECAS|          1802|SANTA EUGENIA|         2|           18029|         29|          77|               7|               3|
0|                2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      17|        5|
|          2|         ARGANZUELA|           205|     DELICIAS|         5|            2062|         62|          71|               6|               6|
0|                2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|        5|
|          9|    MONCLOA-ARAVACA|           907|      ARAVACA|         7|            9083|         83|          55|              15|              14|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      15|        5|
|          3|             RETIRO|           306|   NIÑO JESUS|         6|            3080|         80|          60|               7|               8|
1|                1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|       6|        5|
|          4|          SALAMANCA|           405|        LISTA|         5|            4102|        102|          78|               1|               6|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|       9|        5|
|         20|SAN BLAS-CANILLEJAS|          2001|     SIMANCAS|         1|           20009|          9|    100 o +|               0|               1|
```

• **6.7) Borra esta columna.**

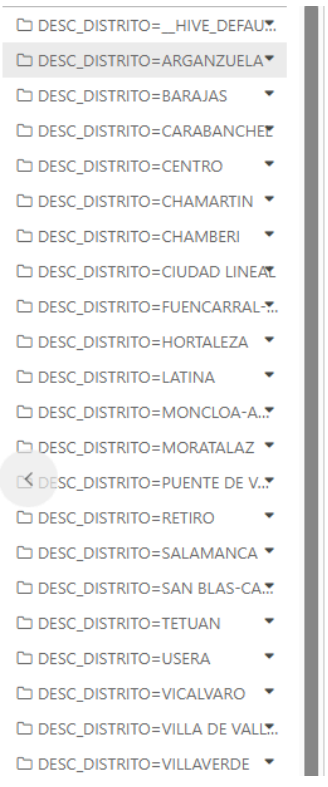
```
%python
df = df.drop("valor_fijo")
df.printSchema()
```

```
df: pyspark.sql.dataframe.DataFrame = [COD_DISTRITC
root
 |-- COD_DISTRITO: string (nullable = true)
 |-- DESC_DISTRITO: string (nullable = true)
 |-- COD_DIST_BARRIO: string (nullable = true)
 |-- DESC_BARRIO: string (nullable = true)
 |-- COD_BARRIO: string (nullable = true)
 |-- COD_DIST_SECCION: string (nullable = true)
 |-- COD_SECCION: string (nullable = true)
 |-- COD_EDAD_INT: string (nullable = true)
 |-- ESPANOLESHOMBRES: string (nullable = true)
 |-- ESPANOLESMUJERES: string (nullable = true)
 |-- EXTRANJEROSHOMBRES: string (nullable = true)
 |-- EXTRANJEROSMUJERES: string (nullable = true)
 |-- FX_CARGA: string (nullable = true)
 |-- FX_DATOS_INI: string (nullable = true)
 |-- FX_DATOS_FIN: string (nullable = true)
 |-- longitud: integer (nullable = true)
```

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

```
%python
# Guardar el DataFrame particionado en Parquet
output_path = "/FileStore/tables/Formacion_Binaia/Practica_Padron/"

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



Se puede ver que ha ido bien, ya que tenemos, los distritos, y dentro de cada uno de ellos, tenemos los barrios dentro de cada distrito.

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

```
%python
from pyspark.sql import SparkSession
df.cache()
df.show()
```

▸ (1) Spark Jobs

```
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|       9|
|          2|       ARGANZUELA|            205|           DELICIAS|        5|            2113|        113|          74|              0|             2|
0|                0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|
|          2|       ARGANZUELA|            206|PALOS DE LA FRONTERA|        6|            2068|         68|          48|              4|             5|
0|                1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|
```

Como estoy trabajando en databricks, no puedo porque el puerto 4040 es local al entorno del cluster de Spark y no está disponible para acceso remoto o desde el navegador local. En Databricks, la administración de Spark y la visualización de trabajos se realiza a través de la interfaz proporcionada por Databricks, no mediante acceso local a puertos.

Podemos visualizar el job de la cache:

Details for Stage 37 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 3 s
Locality Level Summary: Process local: 1
Associated Job Ids: 31

▸ DAG Visualization
▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 1 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	3 s	3 s	3 s	3 s	3 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms

▸ Aggregated Metrics by Executor

Tasks (1)

Show 20 entries   Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Errors
0	101	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-236-246.us-west-2.compute.internal		2024-08-01 10:14:18	3 s		

Showing 1 to 1 of 1 entries   Previous 1 Next

• 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".

```
%python
# Convertir las columnas numéricas a int
df = df.withColumn("ESPANOLESHOMBRES", col("ESPANOLESHOMBRES").cast("int"))
df = df.withColumn("ESPANOLESMUJERES", col("ESPANOLESMUJERES").cast("int"))
df = df.withColumn("EXTRANJEROSHOMBRES", col("EXTRANJEROSHOMBRES").cast("int"))
df = df.withColumn("EXTRANJEROSMUJERES", col("EXTRANJEROSMUJERES").cast("int"))

df.printSchema()
```

Como lo tenia como String las columnas, he hecho un cast para que pueda usarlas como es adecuado.

▸ (1) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [COD_DISTRITO: string, DESC_DISTRITO: string ... 14 more fie

```
+---------------+---------------+-----------------+-----------------+
only showing top 20 rows

root
 |-- COD_DISTRITO: string (nullable = true)
 |-- DESC_DISTRITO: string (nullable = true)
 |-- COD_DIST_BARRIO: string (nullable = true)
 |-- DESC_BARRIO: string (nullable = true)
 |-- COD_BARRIO: string (nullable = true)
 |-- COD_DIST_SECCION: string (nullable = true)
 |-- COD_SECCION: string (nullable = true)
 |-- COD_EDAD_INT: string (nullable = true)
 |-- ESPANOLESHOMBRES: integer (nullable = true)
 |-- ESPANOLESMUJERES: integer (nullable = true)
 |-- EXTRANJEROSHOMBRES: integer (nullable = true)
 |-- EXTRANJEROSMUJERES: integer (nullable = true)
 |-- FX_CARGA: string (nullable = true)
 |-- FX_DATOS_INI: string (nullable = true)
 |-- FX_DATOS_FIN: string (nullable = true)
 |-- longitud: integer (nullable = true)
```

Ejecuto la consulta y vemos los resultados

```
%python
from pyspark.sql.functions import sum, col
# Agrupar por DESC_DISTRITO y DESC_BARRIO, sumar las columnas relevantes
result_df = df.groupBy("DESC_DISTRITO", "DESC_BARRIO").agg(
    sum("ESPANOLESHOMBRES").alias("Total_EspanolesHombres"),
    sum("ESPANOLESMUJERES").alias("Total_EspanolesMujeres"),
    sum("EXTRANJEROSHOMBRES").alias("Total_ExtranjerosHombres"),
    sum("EXTRANJEROSMUJERES").alias("Total_ExtranjerosMujeres")
)

# Ordenar los resultados por Total_ExtranjerosMujeres y Total_ExtranjerosHombres
result_df = result_df.orderBy(col("Total_ExtranjerosMujeres").desc(), col("Total_ExtranjerosHombres").desc())

# Mostrar los resultados
result_df.show()
```

(2) Spark Jobs

result_df: pyspark.sql.dataframe.DataFrame = [DESC_DISTRITO: string, DESC_BARRIO: string ... 4 more fields]

DESC_DISTRITO	DESC_BARRIO	Total_EspanolesHombres	Total_EspanolesMujeres	Total_ExtranjerosHombres	Total_ExtranjerosMujeres
PUENTE DE VALLECAS	SAN DIEGO	14886	16515	9795	9759
CIUDAD LINEAL	PUEBLO NUEVO	23698	27603	7666	8301
LATINA	ALUCHE	24896	29873	7315	8123
CARABANCHEL	VISTA ALEGRE	16076	19626	7580	7988
PUENTE DE VALLECAS	NUMANCIA	17256	19428	7772	7715
VILLAVERDE	VILLAVERDE ALTO C.H.	17489	19502	7354	7379
CENTRO	EMBAJADORES	16514	16102	9510	7121
CIUDAD LINEAL	VENTAS	18949	22299	5552	6381
CARABANCHEL	PUERTA BONITA	12745	14589	6129	6375
LATINA	PUERTA DEL ANGEL	15941	18308	5666	6188
FUENCARRAL-EL PARDO	VALVERDE	27266	29556	5277	5903
CARABANCHEL	SAN ISIDRO	14750	16721	5709	5848
HORTALEZA	VALDEFUENTES	32214	32976	4491	5329
CARABANCHEL	OPAÑEL	11876	14224	4915	5257
VILLAVERDE	LOS ROSALES	14267	15765	5112	5223
VILLA DE VALLECAS	ENSANCHE DE VALLECAS	23888	24623	4167	5061
CENTRO	UNIVERSIDAD	12256	12210	4707	5060

• 6.11) Elimina el registro en caché.

```
%python
df.unpersist()
```

```
Out[36]: DataFrame[COD_DISTRITO: string, DESC_DISTRITO: string, COD_DIST_BARRIO: string, DESC_BARRIO: string, COD_BARRIO: string, COD_DIST_SECCION: string, COD_SECCION: s
tring, COD_EDAD_INT: string, ESPANOLESHOMBRES: int, ESPANOLESMUJERES: int, EXTRANJEROSHOMBRES: int, EXTRANJEROSMUJERES: int, FX_CARGA: string, FX_DATOS_INI: string, FX_DA
TOS_FIN: string, longitud: int]
```

• 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.

```
%python
from pyspark.sql.functions import col, sum
# Suponiendo que 'df' es tu DataFrame original
# Agrupar por DESC_BARRIO y DESC_DISTRITO, sumar ESPANOLESHOMBRES
df_sum = df.groupBy("DESC_BARRIO", "DESC_DISTRITO").agg(
    sum(col("ESPANOLESHOMBRES")).alias("Total_EspanolesHombres")
)

# Mostrar el esquema y las primeras filas para confirmar
df_sum.printSchema()
df_sum.show(10)
```

(2) Spark Jobs

df_sum: pyspark.sql.dataframe.DataFrame = [DESC_BARRIO: string, DESC_D

```
root
 |-- DESC_BARRIO: string (nullable = true)
 |-- DESC_DISTRITO: string (nullable = true)
 |-- Total_EspanolesHombres: long (nullable = true)
```

DESC_BARRIO	DESC_DISTRITO	Total_EspanolesHombres
ACACIAS	ARGANZUELA	15297
VALVERDE	FUENCARRAL-EL PARDO	27266
FUENTELARREINA	FUENCARRAL-EL PARDO	1533
PAVONES	MORATALAZ	3572
EL GOLOSO	FUENCARRAL-EL PARDO	9101
SAN JUAN BAUTISTA	CIUDAD LINEAL	5452
CANILLAS	HORTALEZA	16510
PROSPERIDAD	CHAMARTIN	14458
LEGAZPI	ARGANZUELA	8956
COSTILLARES	CIUDAD LINEAL	9871

only showing top 10 rows

```python
%python
# Unir el DataFrame original con el DataFrame sumado
df_joined = df.join(df_sum, on=["DESC_BARRIO", "DESC_DISTRITO"], how="inner")

# Mostrar el esquema y las primeras filas para verificar la unión
df_joined.printSchema()
df_joined.show(10)
```

(4) Spark Jobs

df_joined: pyspark.sql.dataframe.DataFrame = [DESC_BARRIO: string, DESC_DISTRITO: string ... 15 more fields]

```
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2019|        19|        81|         0|         7|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2018|        18|        34|         7|         4|         2|
1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2088|        88|        59|         2|         2|         1|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2088|        88|        32|         5|        10|         1|
2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2016|        16|        53|        10|        14|         0|
1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2092|        92|        69|        11|        13|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2016|        16|        28|        14|        11|         3|
2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|           2|         202|        2|           2025|        25|        75|         5|         5|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
+-----------+-------------+------------+-----------+----------+---------------+----------+----------+----------+----------+----------+-----
------------+--------------------+-----------+-----------+--------+--------------------+
only showing top 10 rows
```

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

Ventanas (`Window`)

- Propósito: Permiten realizar cálculos avanzados como agregados y rankings dentro de subconjuntos de datos.
- Uso: Definidas con `Window` y aplicadas usando funciones como `SUM()` o `ROW_NUMBER()`.
- Ejemplo: Calcular una suma acumulativa dentro de cada grupo.

Particiones (`partitionBy`)

- Propósito: Organizan datos en disco para mejorar la eficiencia de lectura y escritura.
- Uso: Definidas al escribir datos usando `partitionBy`, dividiendo los datos en subdirectorios.
- Ejemplo: Almacenar datos en diferentes carpetas basadas en valores de columnas.

Diferencia Clave:

- Ventanas: Para cálculos analíticos dentro de datos.
- Particiones: Para organizar datos en almacenamiento distribuido.

```python
%python
from pyspark.sql.window import Window
# Definir la ventana de partición
windowSpec = Window.partitionBy("DESC_BARRIO", "DESC_DISTRITO")

# Crear una nueva columna con la suma total de ESPANOLESHOMBRES por partición
df_with_sum = df.withColumn("Total_EspanolesHombres", sum(col("ESPANOLESHOMBRES")).over(windowSpec))

# Mostrar el esquema y las primeras filas para verificar
df_with_sum.printSchema()
df_with_sum.show(10)
```

(3) Spark Jobs

df_with_sum: pyspark.sql.dataframe.DataFrame = [COD_DISTRITO: string, DESC_DISTRITO: string ... 15 more fields]

```
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|       6|                0|
|          6|       TETUAN|           0|        null|        0|            0|         0|        65|         0|         1|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|       6|                0|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2094|        94|        47|         6|        16|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2108|       108|         5|         2|         4|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2019|        19|        81|         0|         7|         0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2018|        18|        34|         7|         4|         2|
1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2088|        88|        59|         2|         2|         1|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2088|        88|        32|         5|        10|         1|
2|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|          2|   ARGANZUELA|         202|    ACACIAS|        2|           2016|        16|        53|        10|        14|         0|
1|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
+-----------+-------------+------------+-----------+---------+---------------+----------+----------+----------+----------+----------+---
```

```
%python
# Unir el DataFrame original con el DataFrame con la columna de suma
df_joined = df.join(df_with_sum.select("DESC_BARRIO", "DESC_DISTRITO", "Total_EspanolesHombres"),
                    on=["DESC_BARRIO", "DESC_DISTRITO"],
                    how="inner")

# Mostrar el esquema y las primeras filas para verificar la unión
df_joined.printSchema()
df_joined.show(10)
```

▸ (4) Spark Jobs

▸ df_joined: pyspark.sql.dataframe.DataFrame = [DESC_BARRIO: string, DESC_DISTRITO: string ... 15 more fields]

```
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
|    ACACIAS|   ARGANZUELA|          2|          202|        2|           2094|        94|         47|             6|              16|                0|
0|2024-07-07 00:00:...|  2024-07-01|  2024-07-01|      10|               15297|
+-----------+------------+-----------+--------------+---------+---------------+----------+------------+---------------+---------------+-----------------+-----
------------+-------------------+-----------+-----------+--------+---------------------+
only showing top 10 rows
```

• 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:

La función `pivot` en PySpark se usa para reorganizar los datos de una tabla, convirtiendo valores únicos de una columna en nuevas columnas. Permite realizar cálculos agregados, como sumas o promedios, para cada combinación de valores en las columnas de origen.

- `pivot`: Transforma los valores únicos de una columna en encabezados de nuevas columnas.
- Uso común: Crear tablas de contingencia que muestran resultados agregados (por ejemplo, sumas o promedios) organizados en una forma tabular más clara.

Por ejemplo, al pivotar por `DESC_DISTRITO`, cada distrito se convierte en una columna, y se puede calcular la suma de `ESPANOLESMUJERES` para cada `COD_EDAD_INT` en esas nuevas columnas.

```
%python
from pyspark.sql.functions import col, sum
# Filtrar los datos para los distritos específicos
# isin() se utiliza para verificar si los valores en una columna de un DataFrame coinciden con cualquiera de los valores proporcionados en una lista dada. Si un valor en la columna del DataFrame se encuentra en la lista, devuelve verdadero; de lo contrario, devuelve falso. Esta función es útil para filtrar datos basados en valores específicos en los que estás interesado.
df_filtered = df.filter(col("DESC_DISTRITO").isin("CENTRO", "BARAJAS", "RETIRO"))

# Crear la tabla de contingencia (pivot table)
pivot_df = df_filtered.groupBy("COD_EDAD_INT").pivot("DESC_DISTRITO").agg(
    sum("ESPANOLESMUJERES").alias("Total_EspanolesMujeres")
)
pivot_df.show()
```

▸ (7) Spark Jobs

▸ df_filtered: pyspark.sql.dataframe.DataFrame = [COD_DISTRITO: string, DESC_DISTRITO: string ... 13 more fields]

▸ pivot_df: pyspark.sql.dataframe.DataFrame = [COD_EDAD_INT: string, BARAJAS: long ... 2 more fields]

COD_EDAD_INT	BARAJAS	CENTRO	RETIRO
51	449	803	805
7	221	213	416
15	271	255	445
54	372	766	775
11	242	249	435
69	219	579	854
29	170	848	606
42	323	768	731
87	105	265	455
73	260	492	767
64	255	735	838
3	151	181	314
30	194	849	601
34	205	837	602
59	362	791	821
8	239	219	381

• 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.

```
%python
from pyspark.sql.functions import col, round

# Calcular la suma total de ESPANOLESMUJERES para cada rango de edad
pivot_df = pivot_df.withColumn("Total_Sum", col("CENTRO") + col("BARAJAS") + col("RETIRO"))

# Calcular los porcentajes de cada distrito respecto al total y redondear a 2 decimales
pivot_df = pivot_df.withColumn("Centro_Percentage", round((col("CENTRO") / col("Total_Sum")) * 100, 2))
pivot_df = pivot_df.withColumn("Barajas_Percentage", round((col("BARAJAS") / col("Total_Sum")) * 100, 2))
pivot_df = pivot_df.withColumn("Retiro_Percentage", round((col("RETIRO") / col("Total_Sum")) * 100, 2))

# Mostrar el resultado
pivot_df.show()
```

▸ (3) Spark Jobs

▸ pivot_df: pyspark.sql.dataframe.DataFrame = [COD_EDAD_INT: string, BARAJAS: long ... 6 more fields]

COD_EDAD_INT	BARAJAS	CENTRO	RETIRO	Total_Sum	Centro_Percentage	Barajas_Percentage	Retiro_Percentage
51	449	803	805	2057	39.04	21.83	39.13
7	221	213	416	850	25.06	26.0	48.94
15	271	255	445	971	26.26	27.91	45.83
54	372	766	775	1913	40.04	19.45	40.51
11	242	249	435	926	26.89	26.13	46.98
69	219	579	854	1652	35.05	13.26	51.69
29	170	848	606	1624	52.22	10.47	37.32
42	323	768	731	1822	42.15	17.73	40.12
87	105	265	455	825	32.12	12.73	55.15
73	260	492	767	1519	32.39	17.12	50.49
64	255	735	838	1828	40.21	13.95	45.84
3	151	181	314	646	28.02	23.37	48.61
30	194	849	601	1644	51.64	11.8	36.56
34	205	837	602	1644	50.91	12.47	36.62
59	362	791	821	1974	40.07	18.34	41.59
8	239	219	381	839	26.1	28.49	45.41

• 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.

Cada distrito, tendrá sus barrios.

```
%python
# Guardar el DataFrame particionado por DESC_DISTRITO y DESC_BARRIO en un directorio local
df.write.partitionBy("DESC_DISTRITO", "DESC_BARRIO").csv("/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16", header=True)
```

```
%python
# Listar los archivos en el directorio local para verificar la estructura
display(dbutils.fs.ls("/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16"))
```

(3) Spark Jobs

Table

	path
1	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=ARGANZUELA/
2	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=BARAJAS/
3	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=CARABANCHEL/
4	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=CENTRO/
5	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=CHAMARTIN/
6	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=CHAMBERI/
7	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=CIUDAD LINEAL/
8	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=FUENCARRAL-EL PARDO/
9	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=HORTALEZA/
10	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=LATINA/
11	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=MONCLOA-ARAVACA/
12	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=MORATALAZ/
13	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=PUENTE DE VALLECAS/
14	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/DESC_DISTRITO=RETIRO/

Da como esperaba.

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

```
%python
# Guardar el DataFrame particionado por DESC_DISTRITO y DESC_BARRIO en un directorio local en formato Parquet
df.write.partitionBy("DESC_DISTRITO", "DESC_BARRIO").parquet("/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet", mode='overwrite')
```

```
%python
# Listar los archivos en el directorio local para verificar la estructura
display(dbutils.fs.ls("/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet"))
```

(3) Spark Jobs

Table

	path	name	size
1	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=ARGANZUELA/	DESC_DISTRITO=ARGANZUELA/	0
2	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=BARAJAS/	DESC_DISTRITO=BARAJAS/	0
3	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=CARABANCHEL/	DESC_DISTRITO=CARABANCHEL/	0
4	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=CENTRO/	DESC_DISTRITO=CENTRO/	0
5	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=CHAMARTIN/	DESC_DISTRITO=CHAMARTIN/	0
6	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=CHAMBERI/	DESC_DISTRITO=CHAMBERI/	0
7	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=CIUDAD LINEAL/	DESC_DISTRITO=CIUDAD LINEAL/	0
8	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=FUENCARRAL-EL PARDO/	DESC_DISTRITO=FUENCARRAL-EL PARDO/	0
9	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=HORTALEZA/	DESC_DISTRITO=HORTALEZA/	0
10	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=LATINA/	DESC_DISTRITO=LATINA/	0
11	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=MONCLOA-ARAVACA/	DESC_DISTRITO=MONCLOA-ARAVACA/	0
12	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=MORATALAZ/	DESC_DISTRITO=MORATALAZ/	0
13	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=PUENTE DE VALLECAS/	DESC_DISTRITO=PUENTE DE VALLECAS/	0
14	dbfs:/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/DESC_DISTRITO=RETIRO/	DESC_DISTRITO=RETIRO/	0

Ahora compararemos los pesos de los archivos.

```python
%python
# Listar los archivos y subdirectorios en el directorio CSV
csv_dir = "/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16/"
csv_files = dbutils.fs.ls(csv_dir)

# Función para calcular el tamaño total de los archivos en un directorio, incluyendo subdirectorios
def calculate_size(dir_path):
    files = dbutils.fs.ls(dir_path)
    total_size = 0
    for file in files:
        if file.isDir():
            total_size += calculate_size(file.path)
        else:
            total_size += file.size
    return total_size

# Calcular el tamaño total de los archivos CSV
csv_total_size = calculate_size(csv_dir)
print(f"Tamaño total de los archivos CSV: {csv_total_size} bytes")
```

```
Tamaño total de los archivos CSV: 19014515 bytes
```

```python
%python
parquet_dir = "/FileStore/tables/Formacion_Binaia/Practica_Padron/Ej_6_16_parquet/"
parquet_files = dbutils.fs.ls(parquet_dir)

# Calcular el tamaño total de los archivos Parquet
parquet_total_size = calculate_size(parquet_dir)
print(f"Tamaño total de los archivos Parquet: {parquet_total_size} bytes")
```

```
Tamaño total de los archivos Parquet: 6463347 bytes
```

Se puede ver que en parquet, nos ahorramos casi 13 millones de bytes, por lo que es mucho más eficiente, gasta mucha menos memoria y es mucho más rápido para trabajar.

Por otra parte, el código para CSV es más largo porque la estructura de archivos CSV generada por Spark incluye múltiples subdirectorios y archivos para cada partición, requiriendo un recorrido recursivo para calcular su tamaño. En cambio, los archivos Parquet tienen una estructura más compacta y eficiente, lo que simplifica tanto el almacenamiento como el cálculo de su tamaño.

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

Creo sesión spark:

```
✓ 11:40 AM (2s)

%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim, when

spark = SparkSession.builder.appName("Spark y Hive").getOrCreate()
```

Creo la tabla sin corregirla:

```
%python
# Crear la base de datos si no existe
spark.sql("CREATE DATABASE IF NOT EXISTS datos_padron")

# Usar la base de datos
spark.sql("USE datos_padron")

# Crear la tabla padron_txt con formato delimitado
spark.sql("""
    CREATE TABLE IF NOT EXISTS 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 STRING,
        ESPANOLESHOMBRES INT,
        ESPANOLESMUJERES INT,
        EXTRANJEROSHOMBRES INT,
        EXTRANJEROSMUJERES INT,
        FX_CARGA STRING,
        FX_DATOS_INI STRING,
        FX_DATOS_FIN STRING
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ';'
    LINES TERMINATED BY '\n'
    STORED AS TEXTFILE
    TBLPROPERTIES ("skip.header.line.count"="1");
""")
```

Inserto los datos en la tabla, copiando el csv en otra ubicación, porque si ejerzo el comando de “LOAD DATA”, se elimina de la ubicación local:

```
%python
dbutils.fs.cp('/FileStore/tables/Formacion_Binaia/Practica_Padron/padron_estadistica.csv', '/FileStore/tables/Formacion_Binaia/Practica_Padron/padron.csv')
spark.sql("""
    LOAD DATA INPATH '/FileStore/tables/Formacion_Binaia/Practica_Padron/padron.csv'
    INTO TABLE padron_txt
""")
```

Hago la comprobación de que todo ha salido bien:

```
%python
spark.sql("SELECT * FROM padron_txt LIMIT 5").show()
```

▸ (1) Spark Jobs

COD_DISTRITO	DESC_DISTRITO	COD_DIST_BARRIO	DESC_BARRIO	COD_BARRIO	COD_DIST_SECCION	COD_SECCION	COD_EDAD_INT	ESPANOLESHOMBRES	ESPANOLESMUJERES	EXTRANJEROSHOMBRES	EXTRANJEROSMUJERES	FX_CARGA	FX_DATOS_INI	FX_DATOS_FIN
null	DESC_DISTRITO	null	DESC_BARRIO	null	null	null	COD_EDAD_INT	null	null	null	null	FX_CARGA	FX_DATOS_INI	FX_DATOS_FIN
2	"ARGANZUELA ...	202	"ACACIAS ...	2	2094	94	" 47"	6	16	0	0	"2024-07-07 00:00...	"2024-07-01"	"2024-07-01"
18	"VILLA DE VALLECA...	1802	"SANTA EUGENIA ...	2	18029	29	" 77"	7	3	0	2	"2024-07-07 00:00...	"2024-07-01"	"2024-07-01"
2	"ARGANZUELA ...	205	"DELICIAS ...	5	2062	62	" 71"	6	6	0	2	"2024-07-07 00:00...	"2024-07-01"	"2024-07-01"
9	"MONCLOA-ARAVACA ...	907	"ARAVACA ...	7	9083	83	" 55"	15	14	0	0	"2024-07-07 00:00...	"2024-07-01"	"2024-07-01"

Ahora creo la tabla padron_txt_2 mediante CTAS, para crear una tabla sin espacios en blanco y sin comillas dobles:

```
%python
from pyspark.sql.functions import trim, regexp_replace, col

spark.sql("""
    CREATE TABLE IF NOT EXISTS padron_txt_2 AS
    SELECT
        CAST(trim(regexp_replace(COD_DISTRITO, '"', '')) AS INT) AS COD_DISTRITO,
        trim(regexp_replace(DESC_DISTRITO, '"', '')) AS DESC_DISTRITO,
        CAST(trim(regexp_replace(COD_DIST_BARRIO, '"', '')) AS INT) AS COD_DIST_BARRIO,
        trim(regexp_replace(DESC_BARRIO, '"', '')) AS DESC_BARRIO,
        CAST(trim(regexp_replace(COD_BARRIO, '"', '')) AS INT) AS COD_BARRIO,
        CAST(trim(regexp_replace(COD_DIST_SECCION, '"', '')) AS INT) AS COD_DIST_SECCION,
        CAST(trim(regexp_replace(COD_SECCION, '"', '')) AS INT) AS COD_SECCION,
        trim(regexp_replace(COD_EDAD_INT, '"', '')) AS COD_EDAD_INT,
        CAST(trim(regexp_replace(ESPANOLESHOMBRES, '"', '')) AS INT) AS ESPANOLESHOMBRES,
        CAST(trim(regexp_replace(ESPANOLESMUJERES, '"', '')) AS INT) AS ESPANOLESMUJERES,
        CAST(trim(regexp_replace(EXTRANJEROSHOMBRES, '"', '')) AS INT) AS EXTRANJEROSHOMBRES,
        CAST(trim(regexp_replace(EXTRANJEROSMUJERES, '"', '')) AS INT) AS EXTRANJEROSMUJERES,
        trim(regexp_replace(FX_CARGA, '"', '')) AS FX_CARGA,
        trim(regexp_replace(FX_DATOS_INI, '"', '')) AS FX_DATOS_INI,
        trim(regexp_replace(FX_DATOS_FIN, '"', '')) AS FX_DATOS_FIN
    FROM padron_txt;
""")
```

Hago de nuevo comprobación y veo que ha ido bien:

```
spark.sql("SELECT * FROM padron_txt_2 LIMIT 5").show()
```

(2) Spark Jobs

COD_DISTRITO	DESC_DISTRITO	COD_DIST_BARRIO	DESC_BARRIO	COD_BARRIO	COD_DIST_SECCION	COD_SECCION	COD_EDAD_INT	ESPANOLESHOMBRES	ESPANOLESMUJERES	EXTRANJEROSHOMBRES	EXTRANJEROSMUJERES	FX_CARGA	FX_DATOS_INI	FX_DATOS_FIN
null	DESC_DISTRITO	null	DESC_BARRIO	null	null	null	COD_EDAD_INT	null	null	null	null	FX_CARGA	FX_DATOS_INI	FX_DATOS_FIN
2	ARGANZUELA	202	ACACIAS	2	2094	94	47	6	16	0	0	2024-07-07 00:00:...	2024-07-01	2024-07-01
18	VILLA DE VALLECAS	1802	SANTA EUGENIA	2	18029	29	77	7	3	0	2	2024-07-07 00:00:...	2024-07-01	2024-07-01
2	ARGANZUELA	205	DELICIAS	5	2062	62	71	6	6	0	2	2024-07-07 00:00:...	2024-07-01	2024-07-01
9	MONCLOA-ARAVACA	907	ARAVACA	7	9083	83	55	15	14	0	0	2024-07-07 00:00:...	2024-07-01	2024-07-01

Ahora lo hago con expresiones regulares:

```
%python
#expresiones regulares
spark.sql("""
CREATE TABLE padron_hive_regex (
  COD_DISTRITO STRING,
  DESC_DISTRITO STRING,
  COD_DIST_BARRIO STRING,
  DESC_BARRIO STRING,
  COD_BARRIO STRING,
  COD_DIST_SECCION STRING,
  COD_SECCION STRING,
  COD_EDAD_INT STRING,
  ESPANOLESHOMBRES STRING,
  ESPANOLESMUJERES STRING,
  EXTRANJEROSHOMBRES STRING,
  EXTRANJEROSMUJERES STRING,
  FX_CARGA STRING,
  FX_DATOS_INI STRING,
  FX_DATOS_FIN STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex' = '^([0-9]+);\"([^\"]*)\";([0-9]+);\"([^\"]*)\";([0-9]+);([0-9]+);([0-9]+);\"([^\"]*)\";([0-9]*);([0-9]*);([0-9]*);\"([^\"]*)\";\"([^\"]*)\";\"([^\"]*)\"$'
)
STORED AS TEXTFILE;
""")
```

Meto los datos:

Y compruebo que ha salido bien: