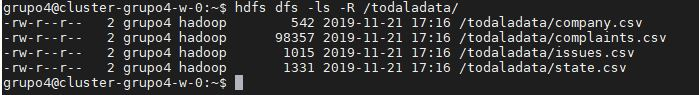
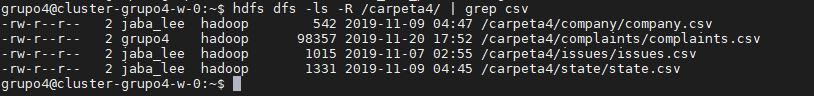
HDFS - HADOOP

Dentro del proyecto tenemos definifos 3 capas de datalake

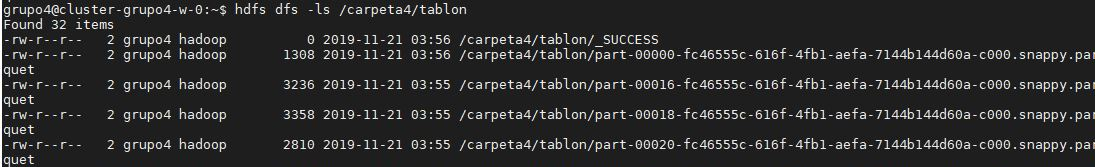
1ra capa



2nda capa



3ra capa



Comandos usados para la creación de las 2 primeras capas

hdfs dfs -mkdir /todaladata/

hdfs dfs -put state.csv /todaladata/

hdfs dfs -put company.csv /todaladata/

hdfs dfs -put issues.csv /todaladata/

hdfs dfs -put complaints.csv /todaladata/

hdfs dfs -cp /todaladata/state.csv /carpeta4/state/

hdfs dfs -cp /todaladata/company.csv /carpeta4/company/

hdfs dfs -cp /todaladata/issues.csv /carpeta4/issues/

hdfs dfs -cp /todaladata/complaints.csv /carpeta4/complaints/

Las carpetas de las 2nda capa fueron creados al momento de crear las tablas

HIVE- BEELINE

Procedemos a crear la base de datos

beeline -u jdbc:hive2://10.128.0.9:10000/

create database trabajo4;

Empezamos por las tablas simples

use trabajo4;

CREATE EXTERNAL TABLE IF NOT EXISTS trabajo4.issues(

id\_issue string COMMENT 'Id\_issue',

issues string COMMENT 'Issues'

)

COMMENT 'Tabla Issues'

ROW FORMAT DELIMITED

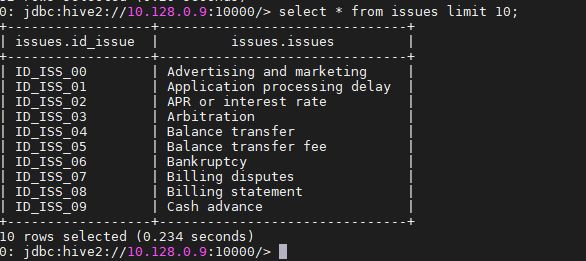
FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/carpeta4/issues'

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



use trabajo4;

CREATE EXTERNAL TABLE IF NOT EXISTS trabajo4.complaints(

date\_received date COMMENT 'Date\_received',

id\_issue string COMMENT 'Id\_issue',

id\_company string COMMENT 'Id\_company',

id\_state string COMMENT 'Id\_state',

consumer\_consent\_provided string COMMENT 'Consumer\_consent\_provided',

submitted\_via string COMMENT 'Submitted\_via',

date\_sent\_to\_company date COMMENT 'Date\_sent\_to\_company',

company\_response\_to\_consumer string COMMENT 'Company\_response\_to\_consumer',

timely\_response string COMMENT 'Timely\_response'

)

COMMENT 'Tabla Complaints'

ROW FORMAT DELIMITED

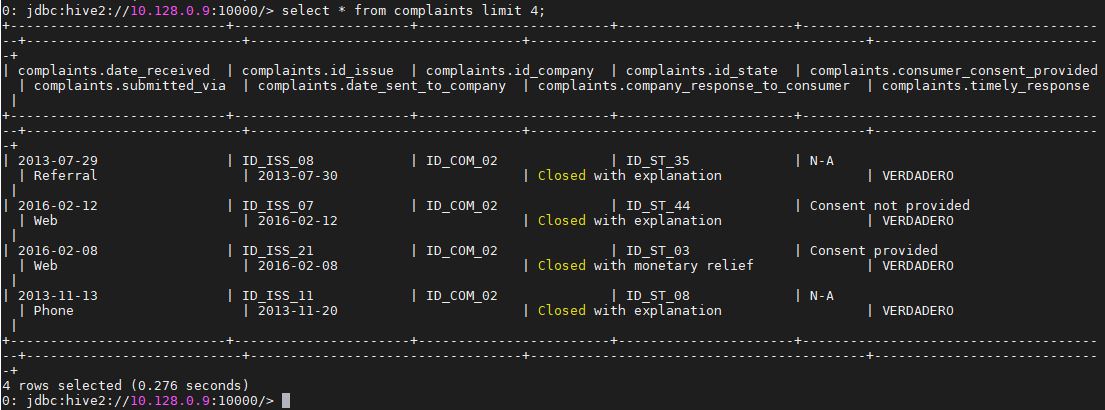
FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/carpeta4/complaints'

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



Procedemos con las tablas estructuradas

use trabajo4;

CREATE EXTERNAL TABLE IF NOT EXISTS trabajo4.company(

id\_company string COMMENT 'Id\_company',

company\_name string COMMENT 'Company\_name',

id\_company\_company\_name struct<id\_company:STRING,company\_name:string> COMMENT 'Id\_company\_company\_name'

)

COMMENT 'Tabla Company'

ROW FORMAT DELIMITED

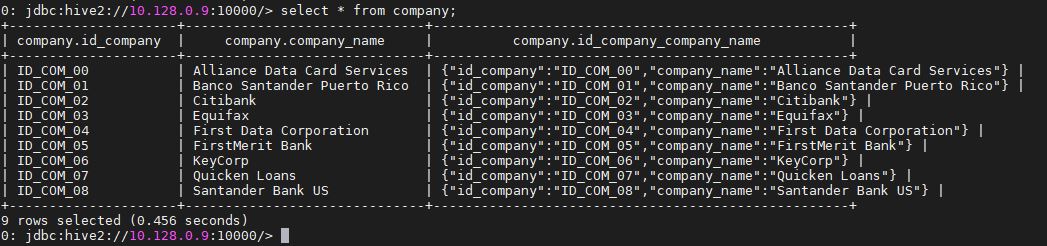
FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

MAP KEYS TERMINATED BY ':'

LOCATION '/carpeta4/company'

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



use trabajo4;

CREATE EXTERNAL TABLE IF NOT EXISTS trabajo4.state(

id\_state string COMMENT 'Id\_state',

state string COMMENT 'State',

id\_state\_state struct<id\_state:STRING,state:string> COMMENT 'Id\_state\_state'

)

COMMENT 'Tabla State'

ROW FORMAT DELIMITED

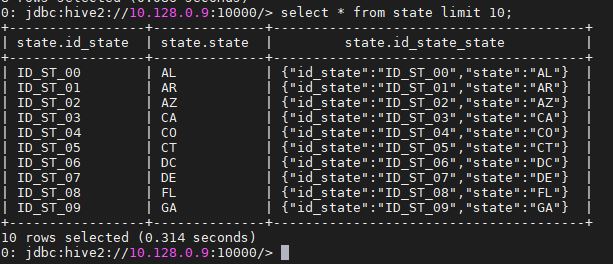
FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

MAP KEYS TERMINATED BY ':'

LOCATION '/carpeta4/state'

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



Luego toca crear una tabla estática y la función que ayudara al llenado de la misma

CREATE EXTERNAL TABLE trabajo4.tabla\_particion\_estatica

(

date\_received date COMMENT 'Date\_received',

id\_issue string COMMENT 'Id\_issue',

id\_company string COMMENT 'Id\_company',

id\_state string COMMENT 'Id\_state',

consumer\_consent\_provided string COMMENT 'Consumer\_consent\_provided',

submitted\_via string COMMENT 'Submitted\_via',

date\_sent\_to\_company date COMMENT 'Date\_sent\_to\_company',

company\_response\_to\_consumer string COMMENT 'Company\_response\_to\_consumer',

timely\_response string COMMENT 'Timely\_response'

)

PARTITIONED BY (mensual STRING)

STORED AS PARQUET

LOCATION '/carpeta4/complaints\_part\_esta/tabla\_particion\_estatica';

insert into trabajo4.tabla\_particion\_estatica partition(mensual = '201211')

select

from\_unixtime (unix\_timestamp(Concat(substring(date\_received,1,4),SUBSTRING(date\_received,5,2), SUBSTRING(date\_received,7,2)), 'yyyyMMdd'), 'yyyy-MM-dd') as date\_received,

id\_issue,

id\_company,

id\_state,

consumer\_consent\_provided,

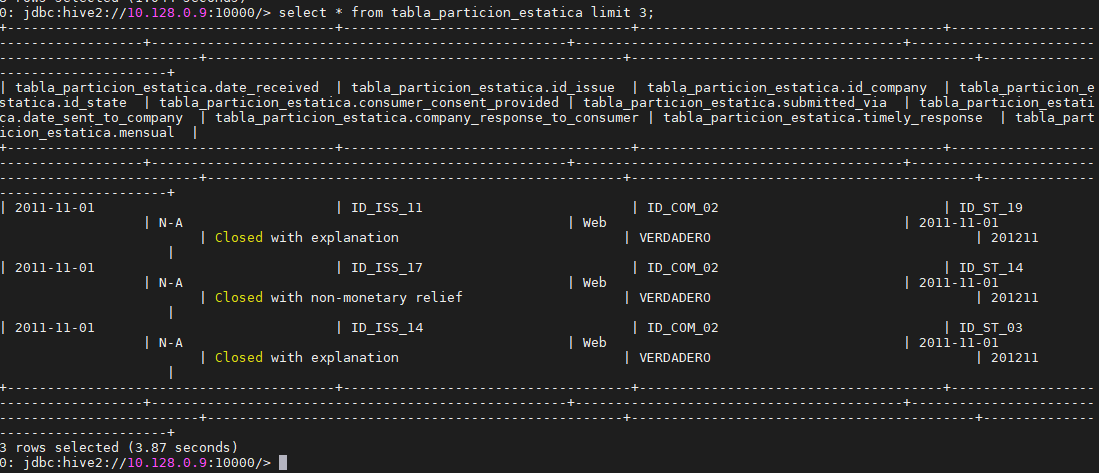
submitted\_via,

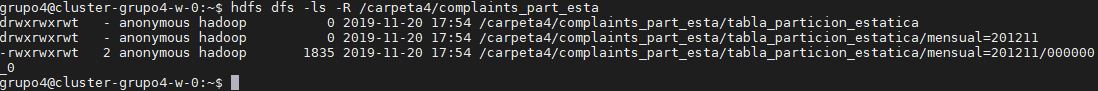
from\_unixtime (unix\_timestamp(Concat(substring(date\_sent\_to\_company,1,4),SUBSTRING(date\_sent\_to\_company,5,2), SUBSTRING(date\_sent\_to\_company,7,2)), 'yyyyMMdd'), 'yyyy-MM-dd') as date\_sent\_to\_company,

company\_response\_to\_consumer,

timely\_response

from trabajo4.complaints where date\_received like '2012-11-%';





Luego toca crear una tabla dinámica y la función que ayudara al llenado de la misma

CREATE EXTERNAL TABLE trabajo4.tabla\_particion\_dinamica

(

id\_issue string COMMENT 'Id\_issue',

id\_company string COMMENT 'Id\_company',

id\_state string COMMENT 'Id\_state',

consumer\_consent\_provided string COMMENT 'Consumer\_consent\_provided',

submitted\_via string COMMENT 'Submitted\_via',

date\_sent\_to\_company date COMMENT 'Date\_sent\_to\_company',

company\_response\_to\_consumer string COMMENT 'Company\_response\_to\_consumer',

timely\_response string COMMENT 'Timely\_response'

)

PARTITIONED BY (date\_received STRING)

STORED AS PARQUET

LOCATION '/carpeta4/complaints\_part\_dina/tabla\_particion\_dinamica';

set hive.exec.dynamic.partition.mode=nonstrict;

insert into trabajo4.tabla\_particion\_dinamica partition(date\_received)

select

id\_issue,

id\_company,

id\_state,

consumer\_consent\_provided,

submitted\_via,

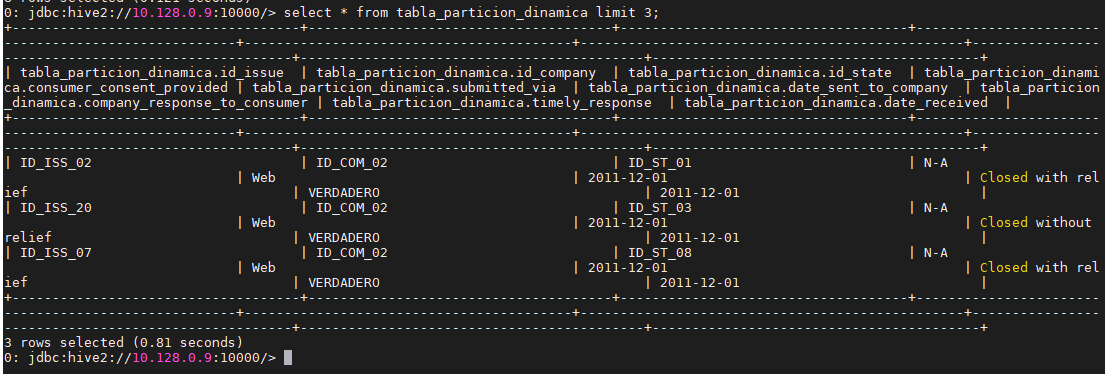
from\_unixtime (unix\_timestamp(Concat(substring(date\_sent\_to\_company,1,4),SUBSTRING(date\_sent\_to\_company,5,2), SUBSTRING(date\_sent\_to\_company,7,2)), 'yyyyMMdd'), 'yyyy-MM-dd') as date\_sent\_to\_company,

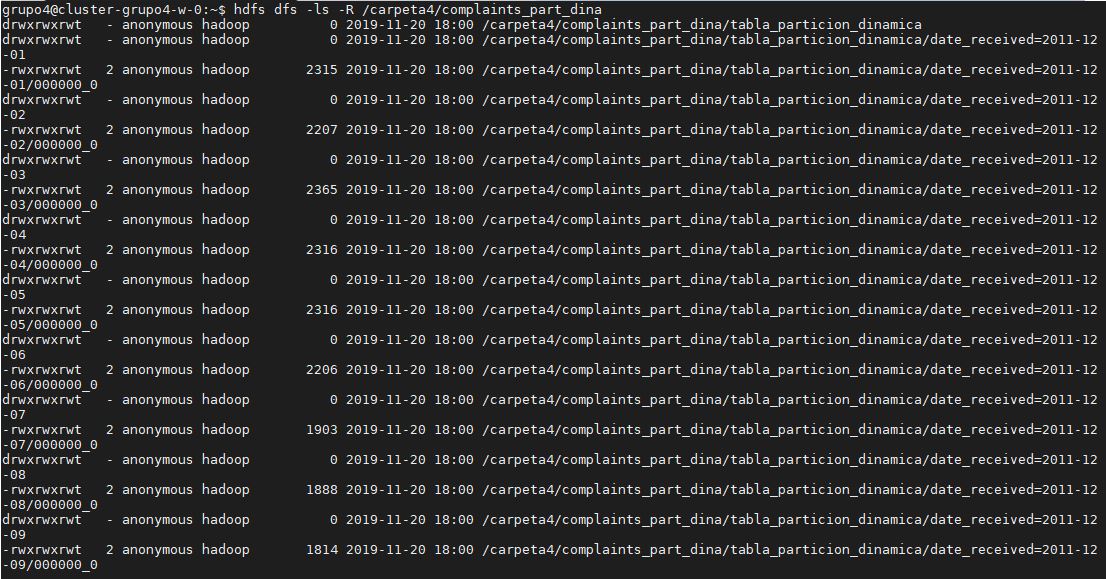
company\_response\_to\_consumer,

timely\_response,

from\_unixtime (unix\_timestamp(Concat(substring(date\_received,1,4),SUBSTRING(date\_received,5,2), SUBSTRING(date\_received,7,2)), 'yyyyMMdd'), 'yyyy-MM-dd') as date\_received

from trabajo4.complaints where date\_received like '2012-0%';





Ahora procedemos a crear el bucking

CREATE EXTERNAL TABLE IF NOT EXISTS trabajo4.company\_buckets (

id\_company string COMMENT 'Id\_company',

company\_name string COMMENT 'Company\_name',

id\_company\_company\_name struct<id\_company:STRING,company\_name:string> COMMENT 'Id\_company\_company\_name'

)

CLUSTERED BY (company\_name) INTO 2 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

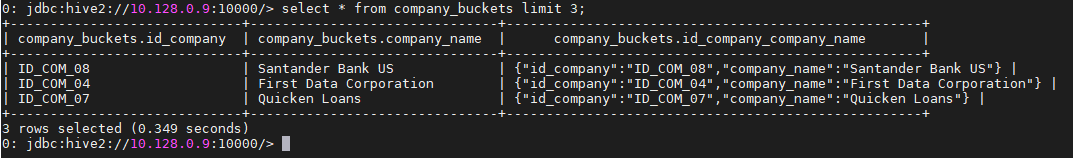
MAP KEYS TERMINATED BY ':'

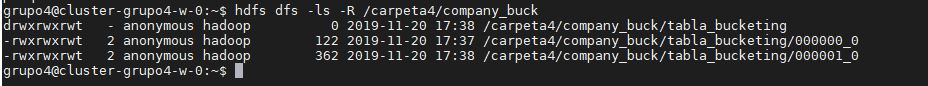
LOCATION '/carpeta4/company\_buck/tabla\_bucketing';

set map.reduce.tasks = 2;

set hive.enforce.bucketing = true;

INSERT OVERWRITE TABLE trabajo4.company\_buckets SELECT \* FROM trabajo4.company;





SPARK

Procedemos a pasar las tablas a data frames

val df\_state = spark.read.table("trabajo4.state")

df\_state.show()

df\_state.printSchema()

val df\_issues = spark.read.table("trabajo4.issues")

df\_issues.show()

df\_issues.printSchema()

val df\_complaints = spark.read.table("trabajo4.complaints")

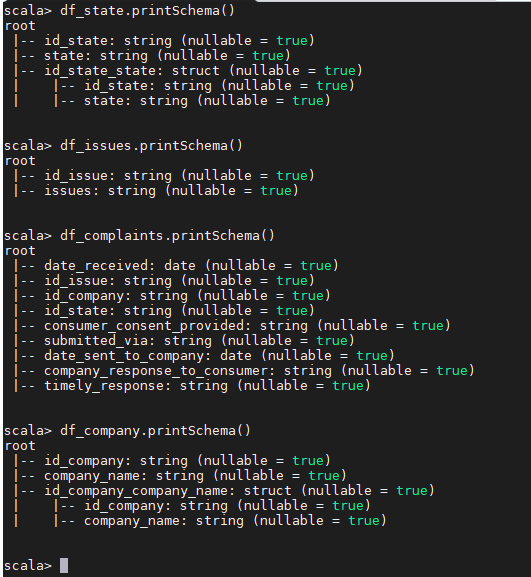
df\_complaints.show()

df\_complaints.printSchema()

val df\_company = spark.read.table("trabajo4.company")

df\_company.show()

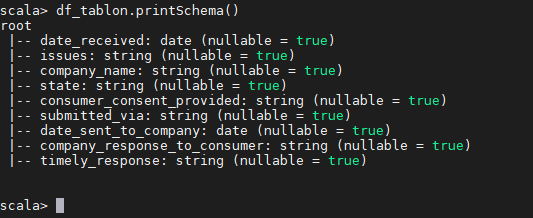
df\_company.printSchema()



val df\_tablon =df\_complaints.as("cot").join(df\_issues.as("iss"),"id\_issue").join(df\_state.as("sta"),"id\_state").join(df\_company.as("coy"),"id\_company").select("cot.date\_received","iss.issues","coy.company\_name","sta.state","cot.consumer\_consent\_provided","cot.submitted\_via","cot.date\_sent\_to\_company","cot.company\_response\_to\_consumer","cot.timely\_response")

df\_tablon.show()

df\_tablon.printSchema()



Guardamos el dataframe en parquet y creamos una tabla con la data a analizar

df\_tablon.write.format("parquet").save("/carpeta4/tablon/")

df\_tablon.createOrReplaceTempView("tablon");

spark.sql("create table if not exists trabajo4.tablon as select \* from tablon");

La “/carpeta4/tablon/" es la tercera capa del datalake que se generará

