**PRACTICA MODULO ARQUITECTURA BIG DATA  
SERAFIN BELIZON GOMEZ**

**DISEÑO**

**DEFINIR ANALÍTICA A REALIZAR**

Construiremos un Data Lake que nos permitirá analizar la actividad de los clientes de KC Telecom S.L., cuyos datos provienen de los operacionales definidos en el módulo de Data101, y de datos socioeconómicos que obtendremos vía web, y que cruzaremos con las direcciones de dichos clientes.

Para ello, definiremos un “Tablón”, cuyos campos serán los indicadores necesarios para realizar dicho análisis, por cada cliente y periodo correspondiente.

Por ejemplo, tomaremos como KPIs el porcentaje del método de pago utilizado, sexo del cliente y cantidad adquirida respecto del total de dichos valores por cada periodo.

Quedando el la estructura del tablón de la siguiente manera:

|  |
| --- |
| **TABLON** |
| id\_cliente |
| periodo |
| ciudad |
| nivel\_se |
| profesion |
| sexo |
| porcentaje\_sexo |
| cantidad\_periodo |
| porcentaje\_cantidad |
| metodo\_pago |
| porcentaje\_metodo\_pago |

Para crear dicha tabla, usaremos la query HIVE:

CREATE TABLE CLIENTES\_KPI (

id\_cliente bigint,

ciudad string,

nivel\_se string,

profesión string,

sexo string,

porcentaje\_sexo doublé,

cantidad\_periodo int,

porcentaje \_cantidad doublé,

método\_pago\_periodo string,

porcentaje \_metodo\_pago double)

PARTITIONED BY (periodo string)

CLUSTERED BY (id\_cliente) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

Y para cargar datos en dicho tablón final:

INSERT OVERWRITE TABLE CLIENTES\_KPI

PARTITION (periodo)

SELECT

CLI.ID\_CLIENTE,

CLI.PERIODO,

CLI.CIUDAD,

CLI.NIVEL\_SE,

CLI.PROFESION,

CLI.SEXO,

(PER.SEXO\_PERIODO / TOT.SEXO\_TOTAL),

CLI.CANTIDAD\_PERIODO,

(CLI.CANTIDAD\_PERIODO / TOT.CANTIDAD\_TOTAL) PROMEDIO\_CANTIDAD,

CLI.METODO\_PAGO,

(PER.METODO\_PAGO\_PERIODO / TOT.METODO\_PAGO\_TOTAL)

FROM TRUSTED\_PERIODO\_CLIENTES CLI

INNER JOIN TRUSTED\_PERIODO\_AUX\_TOTAL TOT ON TOT.PERIODO = CLI.PERIODO

INNER JOIN TRUSTED\_PERIODO\_AUX\_MET\_SEX PER ON PER.PERIODO = CLI.PERIODO AND

PER.METODO\_PAGO = CLI.METODO\_PAGO AND

PER.SEXO = CLI.SEXO

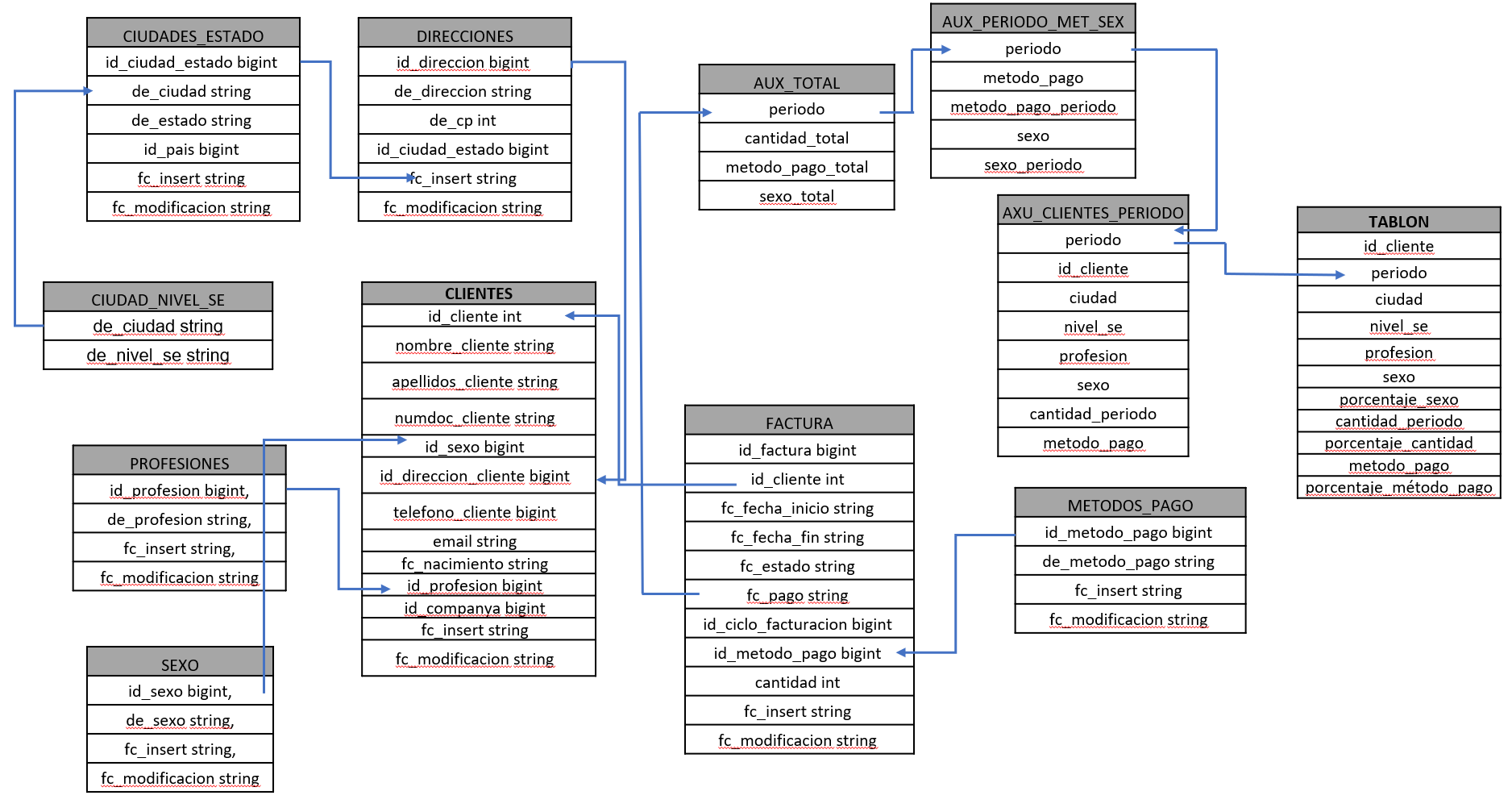
ORDER BY ID\_CLIENTE, PERIODO;

**DEFINIR MODELO LÓGICO PARA ESTE DATAMART**

Del operacional, básicamente necesitaremos datos del cliente, como la dirección, sexo o profesión, así como los datos provenientes de la facturación, como el método de pago o la cantidad.

Por otro lado, para la construcción del tablón, necesitamos de una serie de tablas intermedias que nos permitan realizar los cálculos necesarios para hallar los porcentajes descritos como KPI.

Así que, a nivel lógico, estamos hablando de algo así:



**DEFINIR MODELO FÍSICO PARA HIVE**

Dada la naturaleza de los datos, estas tablas estarán en la zona Trusted de nuestro Datamart, todas como tablas gestionadas hive, formato OCR (debido a la naturaleza “columnar” de los datos que procesaremos) y particionamiento y bucketing según el caso, teniendo en cuenta los JOINs a realizar, así como los datos almacenados en cada tabla.

Las tablas que provienen del operacional tendrían la siguiente estructura:

CREATE TABLE ODS\_TRUSTED\_CLIENTES (

id\_cliente int,

nombre\_cliente string,

apellidos\_cliente string,

numdoc\_cliente string,

id\_sexo bigint,

id\_direccion\_cliente bigint,

telefono\_cliente bigint,

email string,

fc\_nacimiento string,

id\_profesion bigint,

id\_companya bigint,

fc\_insert string,

fc\_modificacion string)

CLUSTERED BY (id\_cliente) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

CREATE TABLE ODS\_TRUSTED\_DIRECCIONES (

id\_direccion bigint,

de\_direccion string,

de\_cp int,

id\_ciudad\_estado bigint,

fc\_insert string,

fc\_modificacion string

)

CLUSTERED BY (id\_direccion) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

CREATE TABLE ODS\_TRUSTED\_CIUDADES\_ESTADOS (

id\_ciudad\_estado bigint,

de\_ciudad string,

id\_pais bigint,

fc\_insert string,

fc\_modificacion string

)

PARTITIONED BY (DE\_ESTADO string)

CLUSTERED BY (id\_ciudad\_estado) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

CREATE TABLE ODS\_TRUSTED\_FACTURAS (

id\_factura bigint,

id\_cliente int,

fc\_fecha\_inicio string,

fc\_fecha\_fin string,

fc\_estado string,

fc\_pago string,

id\_ciclo\_facturacion bigint,

id\_metodo\_pago bigint,

cantidad int,

fc\_insert string,

fc\_modificacion string

)

CLUSTERED BY (id\_factura) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

CREATE TABLE ODS\_TRUSTED\_METODOS\_PAGO (

id\_metodo\_pago bigint,

de\_metodo\_pago string,

fc\_insert string,

fc\_modificacion string

)

CLUSTERED BY (id\_metodo\_pago) INTO 4 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

CREATE TABLE ODS\_TRUSTED\_PROFESIONES (

id\_profesion bigint,

de\_profesion string,

fc\_insert string,

fc\_modificacion string

)

CLUSTERED BY (id\_profesion) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

CREATE TABLE ODS\_TRUSTED\_SEXOS (

id\_sexo bigint,

de\_sexo string,

fc\_insert string,

fc\_modificacion string

)

CLUSTERED BY (id\_sexo) INTO 4 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

CREATE TABLE TRUSTED\_CIUDADES\_SE (

de\_ciudad string,

de\_nivel\_se string)

CLUSTERED BY (de\_ciudad) INTO 5 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS ORC;

Ahora las tablas auxiliares:

CREATE TABLE TRUSTED\_PERIODO\_AUX\_TOTAL (

cantidad\_total bigint,

método\_pago\_total bigint,

sexo\_total bigint,

)

PARTITIONED BY (periodo string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

CREATE TABLE TRUSTED\_PERIODO\_AUX\_MET\_SEX (

metodo\_pago\_periodol bigint,

sexo\_periodo bigint

)

PARTITIONED BY (periodo string, metodo\_pago string, sexo string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

CREATE TABLE TRUSTED\_PERIODO\_CLIENTES (

id\_cliente bigint,

ciudad string,

profesion string,

cantidad\_periodo bigint

)

PARTITIONED BY (periodo string, nivel\_se string, metodo\_pago string, sexo string)

CLUSTERED BY (id\_cliente) INTO 10 BUCKETS

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS OCR;

**IDENTIFICAR FUENTES DE DATOS ESTRUCTURADAS Y “NO ESTRUCTURADAS”**

Las fuentes de datos estructurados provienen de la Base de Datos utilizada en el módulo Data101, que compone el operacional de la empresa cuyos clientes queremos analizar.

Por tanto, lo primero que haríamos sería importar con sqoop dichas tabla, y almacenarlas en HDFS, que será nuestra zona Raw, donde almacenaremos un histórico del estado de cada tabla del operacional por periodo (en nuestro caso, por mes).

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_CLIENTES\_201801 \

--table ODS\_SUM\_CLIENTES --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_DIRECCIONES\_201801 \

--table ODS\_SUM\_DIRECCIONES --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_CIUDADES\_ESTADOS\_201801 \

--table ODS\_SUM\_CIUDADES\_ESTADOS --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_FACTURAS\_201801 \

--table ODS\_SUM\_FACTURAS --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_METODOS\_PAGO\_201801 \

--table ODS\_DM\_METODOS\_PAGO --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_PROFESIONES\_201801 \

--table ODS\_DM\_PROFESIONES --m 1

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_SEXOS\_201801 \

--table ODS\_DM\_SEXOS --m 1

Como, en este caso, no realizaremos ninguna transformación, estos datos no pasaran por Staging, y los almacenaremos directamente en tablas externas de hive.

CREATE EXTERNAL TABLE ODS\_RAW\_CLIENTES\_201801 (

id\_cliente int,

nombre\_cliente string,

apellidos\_cliente string,

numdoc\_cliente string,

id\_sexo bigint,

id\_direccion\_cliente bigint,

telefono\_cliente bigint,

email string,

fc\_nacimiento string,

id\_profesion bigint,

id\_companya bigint,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_CLIENTES\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_CLIENTES\_201801' INTO TABLE ODS\_RAW\_CLIENTES\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_DIRECCIONES\_201801 (

id\_direccion bigint,

de\_direccion string,

de\_cp int,

id\_ciudad\_estado bigint,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_DIRECCIONES\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_DIRECCIONES\_201801' INTO TABLE ODS\_RAW\_DIRECCIONES\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_CIUDADES\_ESTADOS\_201801 (

id\_ciudad\_estado bigint,

de\_ciudad string,

de\_estado string,

id\_pais bigint,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_CIUDADES\_ESTADOS\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_CIUDADES\_ESTADOS\_201801' INTO TABLE ODS\_RAW\_CIUDADES\_ESTADOS\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_FACTURAS\_201801 (

id\_factura bigint,

id\_cliente int,

fc\_fecha\_inicio string,

fc\_fecha\_fin string,

fc\_estado string,

fc\_pago string,

id\_ciclo\_facturacion bigint,

id\_metodo\_pago bigint,

cantidad int,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_FACTURAS\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_FACTURAS\_201801' INTO TABLE ODS\_RAW\_FACTURAS\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_METODOS\_PAGO\_201801 (

id\_metodo\_pago bigint,

de\_metodo\_pago string,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_METODOS\_PAGO\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_METODOS\_PAGO\_201801' INTO TABLE ODS\_RAW\_METODOS\_PAGO\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_PROFESIONES\_201801 (

id\_profesion bigint,

de\_profesion string,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_PROFESIONES\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_PROFESIONES\_201801' INTO TABLE ODS\_RAW\_PROFESIONES\_201801;

CREATE EXTERNAL TABLE ODS\_RAW\_SEXOS\_201801 (

id\_sexo bigint,

de\_sexo string,

fc\_insert string,

fc\_modificacion string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/raw/201801/ODS\_RAW\_SEXOS\_201801';

LOAD DATA INPATH 'raw/201801/ODS\_RAW\_SEXOS\_201801' INTO TABLE ODS\_RAW\_SEXOS\_201801;

Respecto a los datos “no estructurados”, supondremos que queremos obtener el nivel socioeconómico de cada ciudad en la que residen los clientes a partir de una tabla publicada en una web, por lo que construiremos una página html simple para usarla como ejemplo:

<!DOCTYPE html>

<html>

<head>

<style>

table {

font-family: arial, sans-serif;

border-collapse: collapse;

width: 100%;

}

td, th {

border: 1px solid #dddddd;

text-align: left;

padding: 8px;

}

tr:nth-child(even) {

background-color: #dddddd;

}

</style>

</head>

<body>

<table class='table\_SE\_LEVEL'>

<tr>

<th>CITY</th>

<th>SE LEVEL</th>

</tr>

<tr>

<td>LAS VEGAS</td>

<td>HIGH</td>

</tr>

<tr>

<td>RENO</td>

<td>MID</td>

</tr>

<tr>

<td>SACRAMENTO</td>

<td>HIGH</td>

</tr>

<tr>

<td>LOS ANGELES</td>

<td>HIGH</td>

</tr>

<tr>

<td>PHOENIX</td>

<td>HIGH</td>

</tr>

<tr>

<td>PETALUMA</td>

<td>LOW</td>

</tr>

<tr>

<td>STOCKTON</td>

<td>LOW</td>

</tr>

</table>

</body>

</html>

Para recuperar los datos que nos interesan, usaremos la técnica de web scrapping, implementándola en un script de Python.

import urllib2

from bs4 import BeautifulSoup

import csv

# Accedemos a la pagina web (en este caso, local)

page = urllib2.urlopen("file:///DATOS/web/EstadoSE.html").read()

soup = BeautifulSoup(page)

# Localizamos las tablas

all\_tables=soup.find\_all('table')

# Localizamos la tabla que nos hace falta

right\_table=soup.find('table', class\_='wikitable sortable plainrowheaders')

# Escibimos fichero, a partir de cada línea de la tabla

with open("C:\DATOS\web\datos\_SE.csv", "w") as file:

for row in right\_table.findAll("tr"):

cells = row.findAll('td')

if len(cells) > 0:

line = cells[0].find(text=True) + "," + cells[1].find(text=True)

print line

file.write(line)

file.write("\n")

El fichero obtenido, lo copiaremos al servidor con FTP o rcp, por ejemplo, y ya ,desde ahí, lo alojamos en HDFS, para poder ser accesible con hive.

$ hdfs dfs- put datos\_SE.csv /user/hive/stg/datos\_SE.csv

En HIVE, creamos la tabla correspondiente en Staging, y cargamos los datos.

CREATE EXTERNAL TABLE STG\_CIUDADES\_SE\_201801 (

de\_ciudad string,

de\_nivel\_se string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ","

STORED AS TEXTFILE

LOCATION '/hive/stg/datos\_SE.csv’;

LOAD DATA LOCAL INPATH ‘/stg/datos\_SE.csv’ INTO TABLE STG\_CIUDAD\_SE

Por último, para cargar los datos en la zona Trusted

INSERT OVERWRITE TABLE ODS\_TRUSTED\_CLIENTES SELECT \* FROM ODS\_RAW\_CLIENTES\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_DIRECCIONES SELECT \* FROM ODS\_RAW\_DIRECCIONES\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_CIUDADES\_ESTADOS

PARTITION (de\_estado)

SELECT \* FROM ODS\_RAW\_CIUDADES\_ESTADOS\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_FACTURAS SELECT \* FROM ODS\_RAW\_FACTURAS\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_METODOS\_PAGO SELECT \* FROM ODS\_RAW\_METODOS\_PAGO\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_PROFESIONES SELECT \* FROM ODS\_RAW\_PROFESIONES\_201801;

INSERT OVERWRITE TABLE ODS\_TRUSTED\_SEXOS SELECT \* FROM ODS\_RAW\_SEXOS\_201801;

INSERT OVERWRITE TABLE TRUSTED\_CIUDADES\_SE SELECT \* FROM STG\_CIUDADES\_SE\_201801

INSERT OVERWRITE TABLE TRUSTED\_PERIODO\_AUX\_TOTAL

PARTITION (periodo)

SELECT

DATE\_FORMAT(FCT.FC\_PAGO,"%Y%m") PERIODO,

SUM(FCT.CANTIDAD) CANTIDAD\_TOTAL,

COUNT(FCT.ID\_METODO\_PAGO) METODO\_PAGO\_CONT,

COUNT(CLI.ID\_SEXO) SEXO\_CONT

FROM ODS\_TRUSTED\_FACTURAS FCT

INNER JOIN ODS\_TRUSTED\_CLIENTES CLI ON CLI.ID\_CLIENTE = FCT.ID\_CLIENTE

GROUP BY PERIODO;

INSERT OVERWRITE TABLE TRUSTED\_PERIODO\_AUX\_MET\_SEX

PARTITION (periodo, metodo\_pago, sexo)

SELECT

DATE\_FORMAT(FCT.FC\_PAGO,"%Y%m") PERIODO,

MET.DE\_METODO\_PAGO METODO\_PAGO,

COUNT(FCT.ID\_METODO\_PAGO) METODO\_PAGO\_CONT,

SEX.DE\_SEXO SEXO,

COUNT(CLI.ID\_SEXO) SEXO\_CONT

FROM ODS\_TRUSTED\_FACTURAS FCT

INNER JOIN ODS\_TRUSTED\_CLIENTES CLI ON CLI.ID\_CLIENTE = FCT.ID\_CLIENTE

INNER JOIN ODS\_ TRUSTED \_METODOS\_PAGO MET ON MET.ID\_METODO\_PAGO = FCT.ID\_METODO\_PAGO

INNER JOIN ODS\_ TRUSTED \_SEXOS SEX ON SEX.ID\_SEXO = CLI.ID\_SEXO

GROUP BY PERIODO, METODO\_PAGO, SEXO;

INSERT OVERWRITE TABLE TRUSTED\_PERIODO\_CLIENTES

PARTITION (periodo, nivel\_se, metodo\_pago, sexo)

SELECT

CLI.ID\_CLIENTE,

DATE\_FORMAT(FCT.FC\_PAGO,"%Y%m") PERIODO,

CIU.DE\_CIUDAD,

SE.NIVEL\_SE NIVEL\_SE,

PROF.DE\_PROFESION,

SEX.DE\_SEXO,

SUM(FCT.CANTIDAD) CANTIDAD\_PERIODO,

PAG.DE\_METODO\_PAGO METODO\_PAGO\_PERIODO

FROM ODS\_TRUSTED\_CLIENTES CLI

INNER JOIN ODS\_ TRUSTED \_DIRECCIONES DIR ON DIR.ID\_DIRECCION = CLI.ID\_DIRECCION\_CLIENTE

INNER JOIN ODS\_ TRUSTED \_CIUDADES\_ESTADOS CIU ON CIU.ID\_CIUDAD\_ESTADO = DIR.ID\_CIUDAD\_ESTADO

LEFT JOIN TRUSTED\_CIUDADES\_SE SE ON SE.DE\_CIUDAD = CIU.DE\_CIUDAD

INNER JOIN ODS\_ TRUSTED \_PROFESIONES PROF ON PROF.ID\_PROFESION = CLI.ID\_PROFESION

INNER JOIN ODS\_ TRUSTED \_SEXOS SEX ON SEX.ID\_SEXO = CLI.ID\_SEXO

INNER JOIN ODS\_ TRUSTED \_FACTURAS FCT ON FCT.ID\_CLIENTE = CLI.ID\_CLIENTE

INNER JOIN ODS\_ TRUSTED \_METODOS\_PAGO PAG ON PAG.ID\_METODO\_PAGO = FCT.ID\_METODO\_PAGO

GROUP BY CLI.ID\_CLIENTE, PERIODO, NIVEL\_SE, PAG.DE\_METODO\_PAGO;

Ya tendríamos la carga inicial.

Para las cagas periódicas, la estrategia que yo usaría es la de cargar al completo (snapshot) todas las tablas a principio de cada mes, tal cual se ha visto. Y, en cambio, ir actualizando la tabla de facturación semanalmente, por ejemplo.

sqoop import \

--connect jdbc:mysql://192.168.1.46:3306/ODS \

--driver com.mysql.jdbc.Driver \

--username horton --password works \

--target-dir raw/201801/ODS\_RAW\_CIUDADES\_ESTADOS\_201801 \

--table ODS\_SUM\_CIUDADES\_ESTADOS --m 1 \

--incremental append \

--check-column id\_factura \

-last value 573054242