WEATHER EFFECT ON YELP RESTAURANT REVIEWS

1. Context:

In this project we’ll be working to transform Yelp reviews data from JSONS and Los Angeles Weather Data to see if these two things have any relations – we want to investigate the possible impacts of the weather in the quantity of stars of Yelp Reviews.

1. Technologies Used & Process:

We used pure SQL powered by the Snowflake Warehousing Platform.

This project used the Snowflake Client/CLI to get data from JSON and CSV files and send it to a staging space in disk that Snowflake have available, we could also have used AWS S3 or Google Cloud Storage or Blob Storage from Microsoft.

The data was transformed through different layers, where elements were being added. In the staging layer data was loaded as it was coming from the raw files. When sending the data from the Staging Layer to the Operational Data Store Layer a good part of the modeling was done: Primary Keys, Foreign keys, and Data Types were added as results of this modeling – the ERDs will be shared later on this document.

1. Diagram

   Description automatically generatedData Architecture:
2. 1 Screenshots of Created tables in “staging” schema

Graphical user interface, text, application, chat or text message

Description automatically generated

1. 1 Entity Relational Diagram – Operational Data Store

Requirements

Diagram

Description automatically generated

5.2 Entity Relational Diagram – Star Schema Warehouse

Fact and Dimension Tables

Diagram, table

Description automatically generated

6.1 SQL – Modelling from Staging to Operational Data Store

JSONS + CSVS

-- CREATING TABLES FIRST

USE UDACITY\_COURSE;

-- Send csv and json data to ods (define data types, pks, fks, etc)

DROP TABLE IF EXISTS ODS.LA\_PRECIPITATION;

CREATE TABLE ODS.LA\_PRECIPITATION(

date date PRIMARY KEY,

precipitation float,

precipitation\_normal float);

DROP TABLE IF EXISTS ODS.LA\_TEMP;

CREATE TABLE ODS.LA\_TEMP(

date date PRIMARY KEY,

"min" float,

"max" float,

normal\_min float,

normal\_max float);

DROP TABLE IF EXISTS ODS.YELP\_BUSINESS;

CREATE TABLE ODS.YELP\_BUSINESS (

business\_id string PRIMARY KEY,

address string,

categories string,

city string,

hours object,

is\_open string,

latitude float,

longitude float,

name string,

postal\_code string,

review\_count number,

stars number,

state string

);

DROP TABLE IF EXISTS ODS.YELP\_REVIEW;

CREATE TABLE ODS.YELP\_REVIEW (

review\_id TEXT PRIMARY KEY,

business\_id TEXT,

cool NUMBER,

timestamp TIMESTAMP,

date date,

funny number,

stars number,

text TEXT,

useful number,

user\_id TEXT,

FOREIGN KEY (business\_id) REFERENCES ODS.YELP\_BUSINESS(business\_id),

FOREIGN KEY (date) REFERENCES ODS.LA\_TEMP(date),

FOREIGN KEY (date) REFERENCES ODS.LA\_PRECIPITATION(date));

DROP TABLE IF EXISTS ODS.YELP\_USER;

CREATE TABLE ODS.YELP\_USER (

user\_id string PRIMARY KEY,

average\_stars float,

compliment\_cool number,

compliment\_cute number,

compliment\_funny number,

compliment\_hot number,

compliment\_list number,

compliment\_more number,

compliment\_photos number,

compliment\_plain number,

compliment\_profile number,

compliment\_writer number,

cool text,

elite string,

fans text,

friends string,

funny text,

name string,

review\_count number,

useful number,

yelping\_since timestamp

);

DROP TABLE IF EXISTS ODS.YELP\_CHECKING;

CREATE TABLE ODS.YELP\_CHECKING (

check\_id number identity PRIMARY KEY,

business\_id string,

date string,

FOREIGN KEY (business\_id) REFERENCES ODS.YELP\_BUSINESS(business\_id)

);

DROP TABLE IF EXISTS ODS.YELP\_TIP;

CREATE TABLE ODS.YELP\_TIP (

tip\_id number identity PRIMARY KEY,

business\_id string,

compliment\_count number,

date timestamp,

text string,

user\_id string,

FOREIGN KEY (business\_id) REFERENCES ODS.YELP\_BUSINESS(business\_id),

FOREIGN KEY (user\_id) REFERENCES ODS.YELP\_USER(user\_id)

);

DROP TABLE IF EXISTS ODS.YELP\_BUSINESS\_ATTR;

CREATE TABLE ODS.YELP\_BUSINESS\_ATTR (

attribute\_id number identity PRIMARY KEY,

business\_id TEXT,

Alcohol TEXT,

BikeParking TEXT,

BusinessAcceptsCreditCards TEXT,

GoodForDancing TEXT,

HappyHour TEXT,

HasTV TEXT,

NoiseLevel TEXT,

OutdoorSeating TEXT,

RestaurantsGoodForGroups TEXT,

RestaurantsPriceRange TEXT,

RestaurantsReservations TEXT,

WiFi TEXT,

GoodForKids TEXT

);

DROP TABLE IF EXISTS ODS.YELP\_COVID;

CREATE TABLE ODS.YELP\_COVID (

yelp\_covid\_id number IDENTITY,

call\_to\_action\_enabled boolean,

covid\_banner string,

grubhub\_enabled boolean,

request\_quote\_enabled boolean,

temporary\_closed\_until string,

virtual\_services\_offered string,

business\_id string,

delivery\_or\_takeout boolean,

highlights string

);

-- INSERTING DATA

-- inserting data from the csvs that already are in tables in staging to ODS

USE UDACITY\_COURSE;

DELETE FROM ODS.LA\_TEMP WHERE TRUE;

INSERT INTO ODS.LA\_TEMP

SELECT

CONCAT(SUBSTR(date, 1, 4),'-',SUBSTR(date, 5, 2),'-',SUBSTR(date, 7, 2)) as date,

"min",

"max",

normal\_min,

normal\_max

FROM STAGING.LA\_TEMP;

DELETE FROM ODS.LA\_PRECIPITATION WHERE TRUE;

INSERT INTO ODS.LA\_PRECIPITATION

SELECT

CONCAT(SUBSTR(date, 1, 4),'-',SUBSTR(date, 5, 2),'-',SUBSTR(date, 7, 2)) as date,

CASE WHEN precipitation = 'T' THEN 0 ELSE precipitation END,

precipitation\_normal

FROM STAGING.LA\_PRECIPITATION;

-- parsing jsons to flatenned tables

DELETE FROM ODS.YELP\_BUSINESS WHERE TRUE;

INSERT INTO ODS.YELP\_BUSINESS (business\_id, state, address, categories,

city, hours, is\_open, latitude, longitude,

name, postal\_code, review\_count, stars)

SELECT

usersjson: business\_id,

usersjson: state,

usersjson: address,

usersjson: categories,

usersjson: city,

usersjson: hours,

usersjson: is\_open,

usersjson: latitude,

usersjson: longitude,

usersjson: name,

usersjson: postal\_code,

usersjson: review\_count,

usersjson: stars

FROM STAGING.YELP\_ACADEMIC\_DATASET\_BUSINESS;

DELETE FROM ODS.YELP\_REVIEW WHERE TRUE;

INSERT INTO ODS.YELP\_REVIEW(review\_id, business\_id,

timestamp, date, funny,

cool, stars, text, useful, user\_id)

SELECT

usersjson:review\_id,

usersjson:business\_id,

usersjson:date,

date(usersjson:date),

usersjson:funny,

usersjson:cool,

usersjson: stars,

usersjson: text,

usersjson: useful,

usersjson:user\_id

FROM STAGING.YELP\_ACADEMIC\_DATASET\_REVIEW;

DELETE FROM ODS.YELP\_USER WHERE TRUE;

INSERT INTO ODS.YELP\_USER (user\_id, average\_stars, compliment\_cute, compliment\_funny, compliment\_hot,

compliment\_list, compliment\_more, compliment\_photos, compliment\_plain,

compliment\_profile, compliment\_writer, compliment\_cool, elite, fans,

friends, funny, name, review\_count, useful, cool, yelping\_since)

SELECT

usersjson: user\_id,

usersjson: average\_stars,

usersjson: compliment\_cute,

usersjson: compliment\_funny,

usersjson: compliment\_hot,

usersjson: compliment\_list,

usersjson: compliment\_more,

usersjson: compliment\_photos,

usersjson: compliment\_plain,

usersjson: compliment\_profile,

usersjson: compliment\_writer,

usersjson: compliment\_cool,

usersjson: elite,

usersjson: fans,

usersjson: friends,

usersjson: funny,

usersjson: name,

usersjson: review\_count,

usersjson: useful,

usersjson: cool,

usersjson: yelping\_since

FROM STAGING.YELP\_ACADEMIC\_DATASET\_USER;

DELETE FROM ODS.YELP\_CHECKING WHERE TRUE;

INSERT INTO ODS.YELP\_CHECKING (business\_id, date)

SELECT

usersjson: business\_id,

usersjson: date

FROM STAGING.YELP\_ACADEMIC\_DATASET\_CHECKIN;

DELETE FROM ODS.YELP\_TIP WHERE TRUE;

INSERT INTO ODS.YELP\_TIP (business\_id, compliment\_count, date, text, user\_id)

SELECT

usersjson: business\_id as business\_id,

usersjson: compliment\_count as compliment\_count,

usersjson: date as date,

usersjson: text as text,

usersjson: user\_id as user\_id

FROM STAGING.YELP\_ACADEMIC\_DATASET\_TIP;

DELETE FROM ODS.YELP\_BUSINESS\_ATTR WHERE TRUE;

INSERT INTO ODS.YELP\_BUSINESS\_ATTR (business\_id, Alcohol, BikeParking, BusinessAcceptsCreditCards,

GoodForDancing, HappyHour, HasTV, NoiseLevel,

OutdoorSeating,RestaurantsGoodForGroups, RestaurantsPriceRange,

RestaurantsReservations, WiFi, GoodForKids)

SELECT

usersjson: business\_id,

usersjson: attributes.Alcohol,

usersjson: attributes.BikeParking,

usersjson: attributes.BusinessAcceptsCreditCards,

usersjson: attributes.GoodForDancing,

usersjson: attributes.HappyHour,

usersjson: attributes.HasTV,

usersjson: attributes.NoiseLevel,

usersjson: attributes.OutdoorSeating,

usersjson: attributes.RestaurantsGoodForGroups,

usersjson: attributes.RestaurantsPriceRange2,

usersjson: attributes.RestaurantsReservations,

usersjson: attributes.WiFi,

usersjson: attributes.GoodForKids

FROM STAGING.YELP\_ACADEMIC\_DATASET\_BUSINESS;

DELETE FROM ODS.YELP\_COVID WHERE TRUE;

INSERT INTO ODS.YELP\_COVID(call\_to\_action\_enabled, covid\_banner, grubhub\_enabled, request\_quote\_enabled,

temporary\_closed\_until, virtual\_services\_offered, business\_id, delivery\_or\_takeout, highlights)

SELECT

usersjson: "Call To Action enabled",

usersjson: "Covid Banner",

usersjson: "Grubhub enabled",

usersjson: "Request a Quote Enabled",

usersjson: "Temporary Closed Until",

usersjson: "Virtual Services Offered",

usersjson: "business\_id",

usersjson: "delivery or takeout",

usersjson: "highlights"

FROM STAGING.YELP\_ACADEMIC\_DATASET\_COVID\_FEATURES;

6.2 Size of Files (Raw, Raw in Staging – Compressed, in ODS)

Table

Description automatically generated

6.3 SQL – Modelling from ODS to Star Schema Warehouse

-- CREATING STAR SCHEMA TABLES

DROP TABLE IF EXISTS WAREHOUSE.DIM\_DATES;

CREATE TABLE WAREHOUSE.DIM\_DATES(

date DATE PRIMARY KEY,

day\_of\_week INT,

day INT,

month INT,

quarter INT,

year INT);

DROP TABLE IF EXISTS WAREHOUSE.DIM\_BUSINESS;

CREATE TABLE WAREHOUSE.DIM\_BUSINESS (

business\_id TEXT PRIMARY KEY,

name TEXT,

categories TEXT,

state TEXT,

address TEXT,

city TEXT,

is\_open BOOLEAN,

latitude FLOAT,

longitude FLOAT,

postal\_code TEXT,

has\_wifi\_or\_not TEXT

);

DROP TABLE IF EXISTS WAREHOUSE.DIM\_USER;

CREATE TABLE WAREHOUSE.DIM\_USER (

user\_id TEXT PRIMARY KEY,

name TEXT,

average\_stars FLOAT,

yelping\_since TIMESTAMP

);

DROP TABLE IF EXISTS WAREHOUSE.FCT\_YELP\_REVIEWS;

CREATE TABLE WAREHOUSE.FCT\_YELP\_REVIEWS (

business\_id TEXT,

review\_id TEXT,

user\_id TEXT,

timestamp TIMESTAMP,

stars INT,

text TEXT,

max\_temp FLOAT,

min\_temp FLOAT,

max\_temp\_normal FLOAT,

min\_temp\_normal FLOAT,

precipitation FLOAT,

precipitation\_normal FLOAT,

date DATE,

PRIMARY KEY (business\_id, review\_id, timestamp)

);

-- INSERTING DATA FROM ODS TO WAREHOUSE ENVIRONMENT

DELETE FROM WAREHOUSE.DIM\_DATES WHERE TRUE;

INSERT INTO WAREHOUSE.DIM\_DATES(date, day, day\_of\_week,month, quarter, year)

SELECT

date,

EXTRACT('day', date),

EXTRACT('dayofweek',date),

EXTRACT('month', date),

EXTRACT('quarter', date),

EXTRACT('year', date)

FROM ODS.LA\_TEMP;

DELETE FROM WAREHOUSE.DIM\_USER WHERE TRUE;

INSERT INTO WAREHOUSE.DIM\_USER(user\_id, name,

average\_stars, yelping\_since)

SELECT DISTINCT

user\_id,

name,

average\_stars,

yelping\_since

FROM ODS.YELP\_USER;

DELETE FROM WAREHOUSE.DIM\_BUSINESS WHERE TRUE;

INSERT INTO WAREHOUSE.DIM\_BUSINESS(business\_id, name, categories,

state, address, city, is\_open, latitude, longitude,

postal\_code, has\_wifi\_or\_not)

SELECT DISTINCT

t1.business\_id,

name,

categories,

state,

address,

city,

is\_open,

latitude,

longitude,

postal\_code,

CASE

WHEN t2.wifi like '%no%' THEN 'no'

WHEN t2.wifi like '%free%' THEN 'free'

WHEN t2.wifi like '%paid%' THEN 'paid'

ELSE NULL END

FROM ODS.YELP\_BUSINESS t1

JOIN ODS.YELP\_BUSINESS\_ATTR t2

ON t1.business\_id = t2.business\_id ;

DELETE FROM WAREHOUSE.FCT\_YELP\_REVIEWS WHERE TRUE;

DELETE FROM WAREHOUSE.FCT\_YELP\_REVIEWS WHERE TRUE;

INSERT INTO WAREHOUSE.FCT\_YELP\_REVIEWS(review\_id, business\_id,

user\_id, timestamp, stars, text,

max\_temp, min\_temp, max\_temp\_normal,

min\_temp\_normal, precipitation,

precipitation\_normal, date)

SELECT

t1.review\_id,

t1.business\_id,

t1.user\_id,

t1.date,

t1.stars,

t1.text,

t2."max",

t2."min",

t2.normal\_min,

t2.normal\_max,

t3.precipitation,

t3.precipitation\_normal,

t1.date

FROM ODS.YELP\_REVIEW t1

LEFT JOIN ODS.LA\_TEMP AS t2

ON t1.date = t2.date

LEFT JOIN ODS.LA\_PRECIPITATION AS t3

ON t1.date = t3.date;

6.4 SQL Code – View for Reporting, Analytics-Ready Data

-- VIEW WITH EVERYTHING, NOT SUMMARIZED

SELECT

t1.review\_id,

t1.timestamp,

t1.date AS rev\_date,

t1.stars AS rev\_stars,

t1.text AS rev\_text,

t1.max\_temp,

t1.min\_temp,

t1. max\_temp\_normal,

t1.min\_temp\_normal,

t1.precipitation,

t1.precipitation\_normal,

buz.name AS buz\_name,

buz.categories AS buz\_categ,

buz.state AS buz\_state,

buz.address AS buz\_address,

buz.is\_open AS buz\_is\_open,

buz.latitude AS buz\_lat,

buz.longitude AS buz\_long,

buz.postal\_code AS buz\_postal\_code,

buz.has\_wifi\_or\_not AS buz\_has\_wifi,

user.name AS user\_name,

user.average\_stars AS user\_avg\_stars,

user.yelping\_since AS user\_since,

dates.day\_of\_week,

dates.day,

dates.month,

dates.quarter,

dates.year

FROM WAREHOUSE.FCT\_YELP\_REVIEWS t1

LEFT JOIN WAREHOUSE.DIM\_BUSINESS AS buz

ON t1.business\_id = buz.business\_id

LEFT JOIN WAREHOUSE.DIM\_DATES AS dates

ON t1.date = dates.date

LEFT JOIN WAREHOUSE.DIM\_USER AS user

ON t1.user\_id = user.user\_id;

-- GROUPED VIEW, SUMMARIZATION, AVG STARS, TEMPERATURE AND PRECIPITATION

SELECT

concat(dates.year,'-',dates.month) as year\_month,

count(review\_id) as number\_reviews,

avg(t1.stars) AS avg\_stars,

(avg(max\_temp)+avg(min\_temp)/2) AS avg\_temp,

avg(t1.precipitation) AS avg\_precipitation

FROM WAREHOUSE.FCT\_YELP\_REVIEWS t1

LEFT JOIN WAREHOUSE.DIM\_BUSINESS AS buz

ON t1.business\_id = buz.business\_id

LEFT JOIN WAREHOUSE.DIM\_DATES AS dates

ON t1.date = dates.date

LEFT JOIN WAREHOUSE.DIM\_USER AS user

ON t1.user\_id = user.user\_id

GROUP BY concat(dates.year,'-',dates.month)

HAVING number\_reviews > 10

ORDER BY year\_month DESC;