WEATHER EFFECT ON YELP RESTAURANT REVIEWS

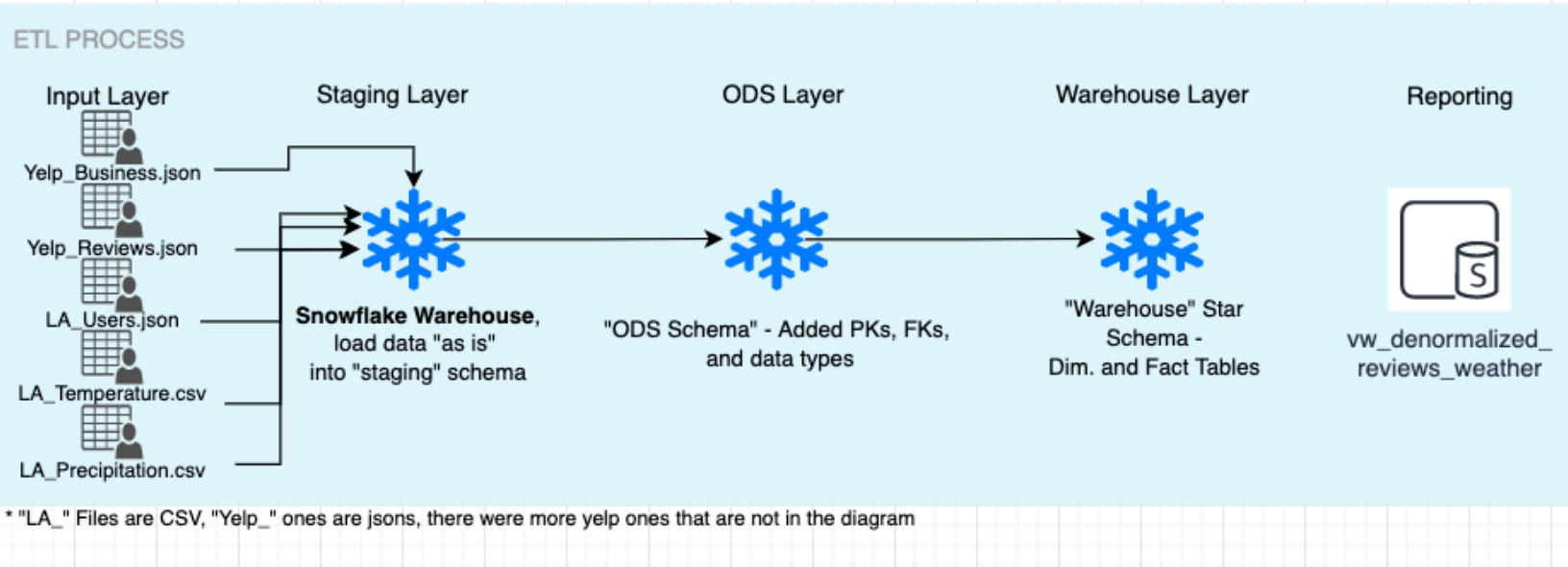
1. Context:

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.

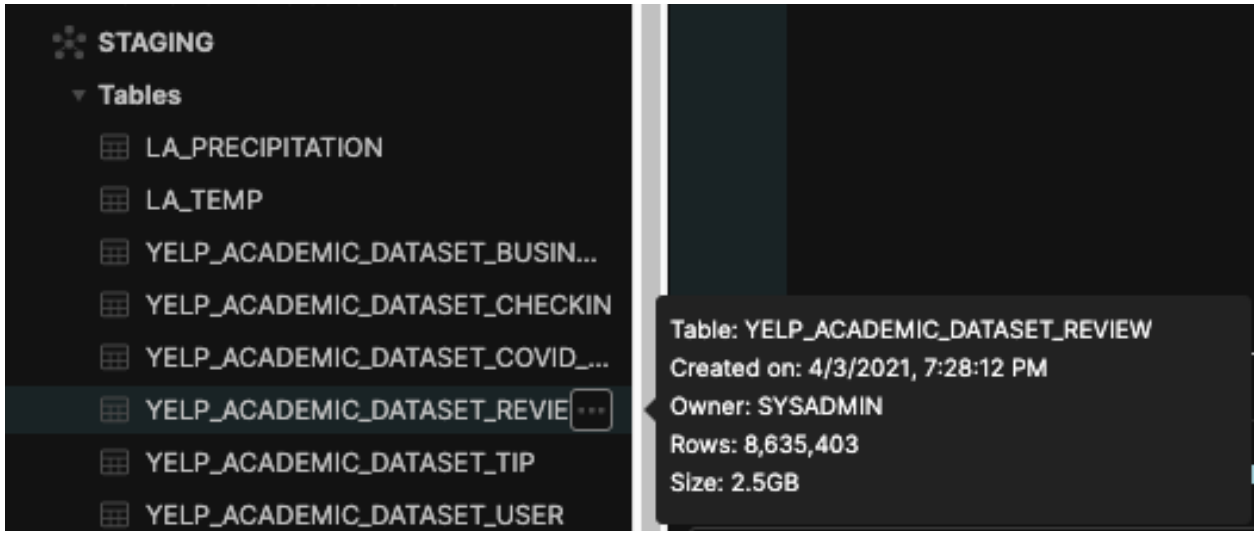
2. Technologies Used & Process:

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.

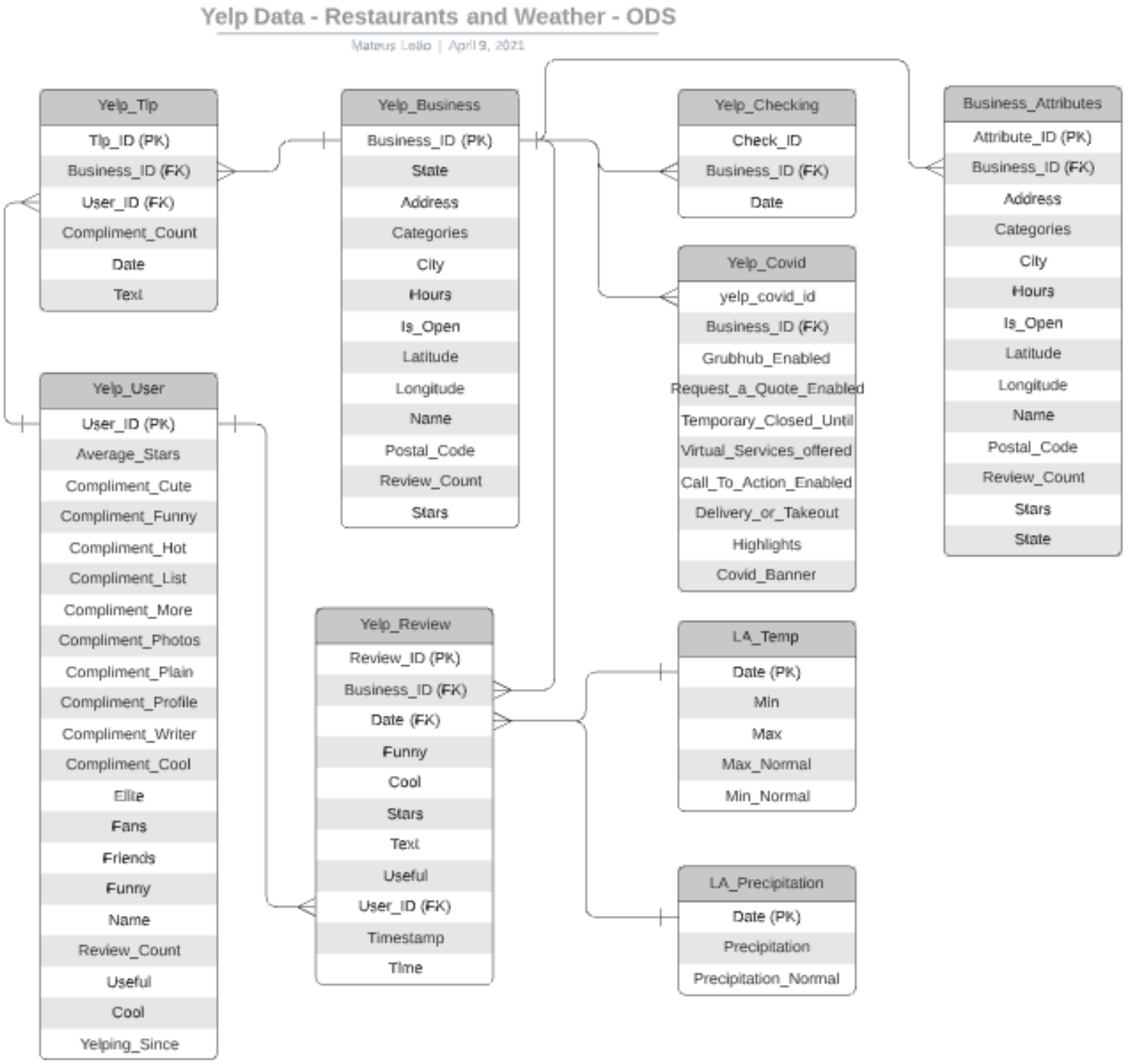
3. Data Architecture:



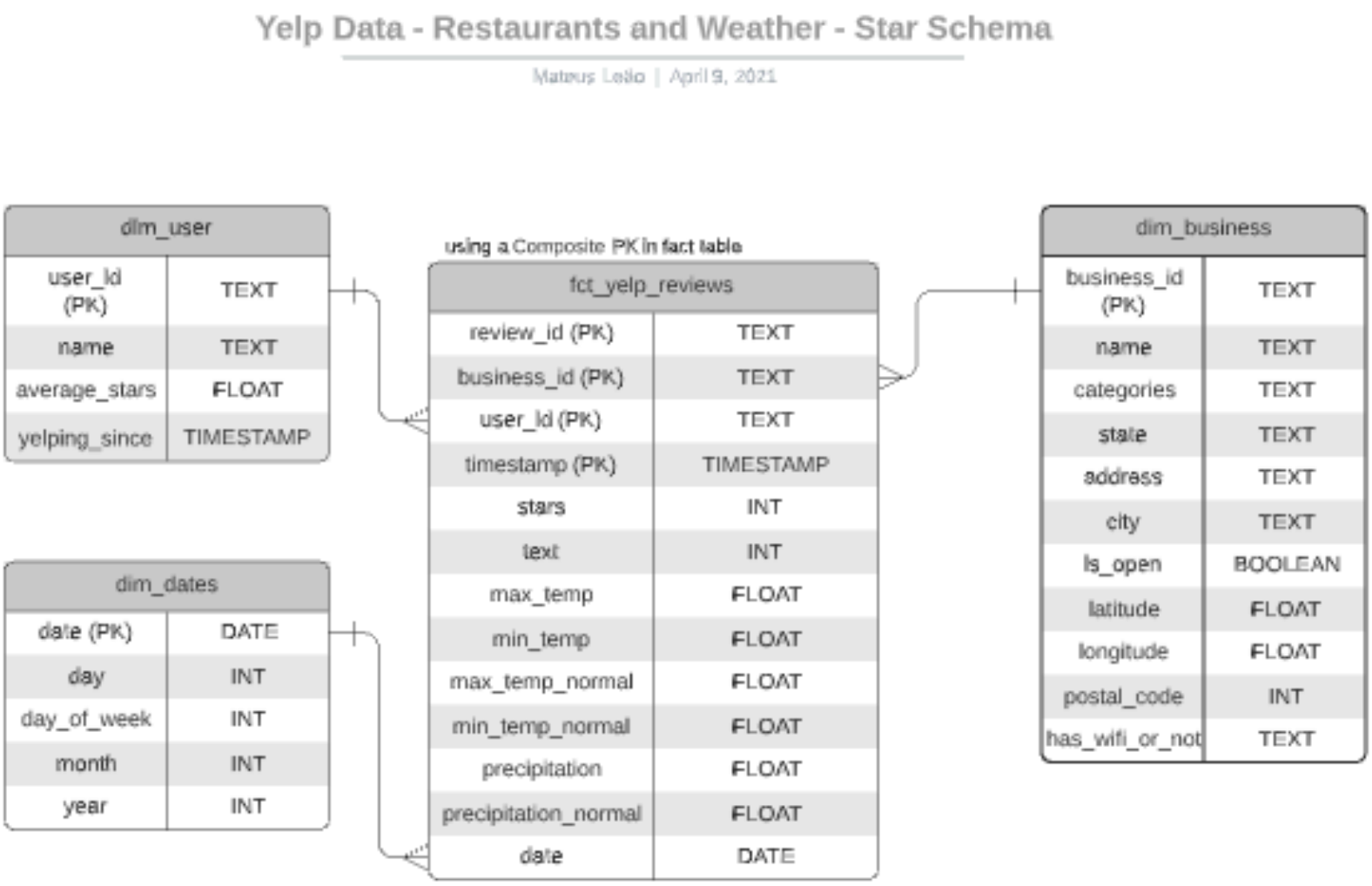
4. 1 Screenshots of Created tables in “staging” schema



5. 1 Entity Relational Diagram – Operational Data Store Requirements



5.2 Entity Relational Diagram – Star Schema Warehouse Fact and Dimension Tables



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)

File_Name	File Type	Raw Files Si	Size in Stagi	Size in ODS
LA_Precipitation.csv	csv	1.1mb	0.233mb	0.262mb
LA_Temp.csv	csv	1.6mb	0.347 mb	0.313mb
Yelp_Business	json	124.4mb	21.16mb	10.8mb
Yelp_Checking	json	398.9mb	109.61mb	109.7mb
Yelp_COVID	json	64.8mb	6,95mb	5.7mb
Yelp_Review	json	6940mb	2638mb	2600mb
Yelp_Tip	json	230.5mb	91,87mb	76mb
Yelp_User	json	3740mb	2151mb	2000mb

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;
```