Project: Predictive Analytics Capstone

1. Introduction

The objective of this project is to use actual YELP and climate datasets in order to analyze the effects the weather has on customer reviews of restaurants. The data for temperature and precipitation observations are from the Global Historical Climatology Network-Daily (GHCN-D) database. A leading industry cloud-native data warehouse system Snowflake will be used for all aspects of the project to demonstrate the ability to design data systems and design a Data Warehouse DWH for the purpose of reporting and online analytical processing (OLAP).

2. Data Architecture Diagram

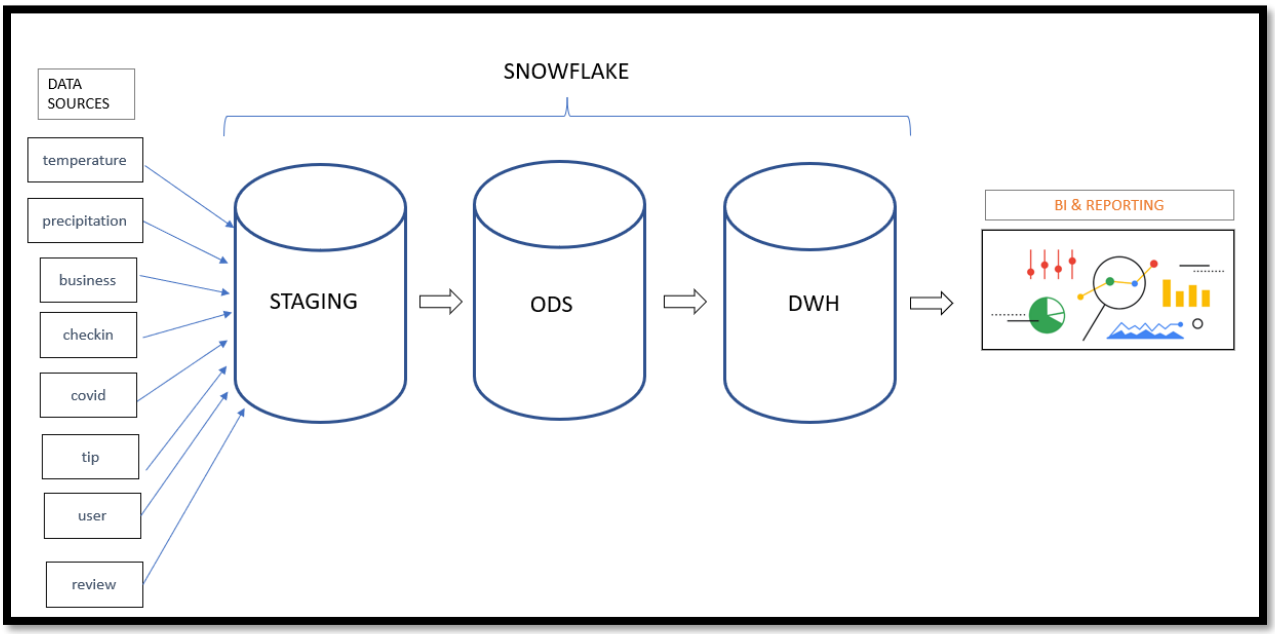


Figure 01 – Data architecture diagram for the project

The data architecture diagram can be explained as follows -

Data Sources:

1. 8 data files are downloaded in the local disc, these are-

    a. **JSON files:** the JSON files are Yelp data

- Business
- Check-in
- Covid
- Review
- Tip
- Users

b. **2 CSV files:** contains weather data

- Temperature
- Precipitation

2. All the files will be then uploaded to the staging schema.

3. From staging schema, the data will be copied and converted to appropriate data format and will placed into staging schema.

4. From staging schema the data will be copied to DWH for the analytical processing environment.

3. STAGING

3.1 STAGING SCHEMA

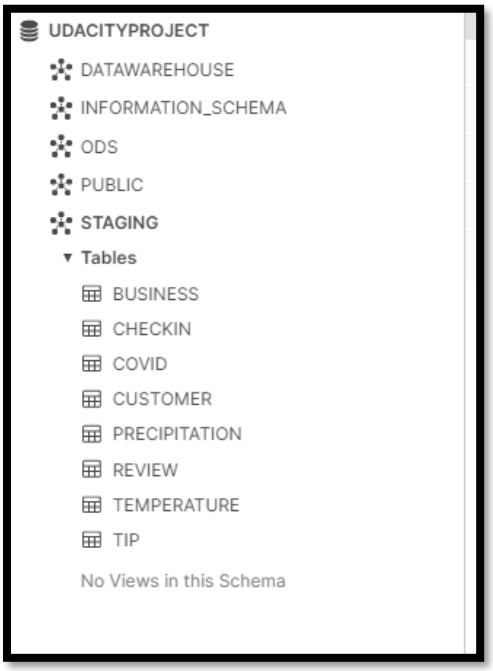


Figure 02 – Shows the tables created in the staging schema to upload the data. Screenshot of 6 tables created upon upload of YELP data.

Business, Review, Tip, Customer, Checkin and Covid tables have been created by loading local data files into a Snowflake staging schema, using the command-line snowsql tool. The code is documented in Appendix -1.

3.2 BUSINESS table:

```
SELECT *
FROM UDACITYPROJECT.STAGING.BUSINESS
LIMIT 5;
```

Results | Data Preview

Query ID | SQL | 1.26s | 5 rows

Filter result... | Copy

Row	BUSINESSJSON
1	{ "address": "1616 Chapala St, Ste 2", "attributes": { "ByAppointmentOnly": "True" }, "business_id": "Pns2l4eNsfO8kk83dixA6A", "categories": "Doctors, Traditional Chinese Medicine, Naturopathic/Holistic, Acupuncture, Health & Medical, Nutr...
2	{ "address": "87 Grasso Plaza Shopping Center", "attributes": { "BusinessAcceptsCreditCards": "True" }, "business_id": "mpf3x-BjTdTEA3yCZrAYPw", "categories": "Shipping Centers, Local Services, Notaries, Mailbox Centers, Printing Services"...
3	{ "address": "5255 E Broadway Blvd", "attributes": { "BikeParking": "True", "BusinessAcceptsCreditCards": "True", "BusinessParking": "{'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False}", "ByAppointmen...
4	{ "address": "935 Race St", "attributes": { "Alcohol": "u'none'", "BikeParking": "True", "BusinessAcceptsCreditCards": "False", "BusinessParking": "{'garage': False, 'street': True, 'validated': False, 'lot': False, 'valet': False}",...
5	{ "address": "101 Walnut St", "attributes": { "BikeParking": "True", "BusinessAcceptsCreditCards": "True", "BusinessParking": "{'garage': None, 'street': None, 'validated': None, 'lot': True, 'valet': False}", "Caters": "False", ...

Figure 03 – Shows first 05 rows of the business table created in the staging schema

3.3 CHECKIN Table:

```
SELECT *
FROM UDACITYPROJECT.STAGING.CHECKIN
LIMIT 5;
```

Results | Data Preview

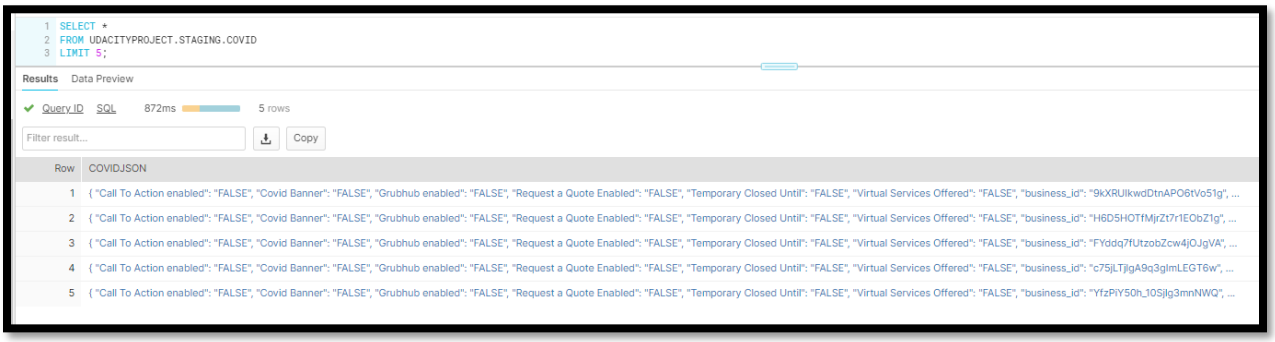
Query ID | SQL | 970ms | 5 rows

Filter result... | Copy

Row	CHECKINJSON
1	{ "business_id": "MuCoKxrahsUHyY0pXcTI6Q", "date": "2013-07-29 01:50:47, 2014-06-11 01:44:16, 2014-10-18 01:04:18, 2014-11-14 21:02:20, 2015-01-03 22:34:09, 2015-01-12 23:44:29, 2015-02-01 20:36:40, 2015-06-12 22:08:36, 2015-09-05 21:32:31, 2015-11-...
2	{ "business_id": "MuDZTdhx3WhWi-ciLvZ-nQ", "date": "2013-01-05 00:33:20, 2013-02-21 01:10:08, 2013-08-18 20:19:44, 2013-09-05 22:19:36, 2014-02-01 19:52:19, 2014-03-23 20:54:05, 2014-04-29 20:39:19, 2014-05-14 21:04:51, 2014-08-29 20:38:11, 2014-08-...
3	{ "business_id": "MuE4wrLB78kb47ASYIlajQ", "date": "2012-03-31 16:48:55, 2012-03-31 17:27:10, 2012-03-31 18:07:48, 2012-03-31 18:16:10, 2012-03-31 18:24:59, 2012-03-31 18:30:21, 2012-03-31 18:47:24, 2012-03-31 19:14:49, 2012-06-11 02:49:25, 2013-03-...
4	{ "business_id": "MuJACN0_RvP2VKT-M51SzA", "date": "2017-11-15 21:07:06, 2018-01-28 19:23:24, 2018-01-28 19:32:55, 2018-03-20 20:26:25, 2018-03-24 22:08:25, 2018-03-24 22:36:14, 2018-05-02 23:20:36, 2018-05-27 18:45:36, 2018-06-22 18:00:34, 2018-07-...
5	{ "business_id": "MuJBELBk9VXJjH0tryDh1w", "date": "2015-09-30 17:14:16, 2015-10-02 19:46:51, 2015-10-06 16:48:50, 2015-10-15 18:33:23, 2015-11-06 01:01:09, 2015-11-10 19:08:35, 2015-11-13 18:14:37, 2015-11-20 19:07:39, 2015-12-13 01:12:46, 2015-12-...

Figure 04 – Shows first 05 rows of the checkin table created in the staging schema

3.4 COVID Table:



```
SELECT *
FROM UDACITYPROJECT.STAGING.COVID
LIMIT 5;
```

Results | Data Preview

Query ID | SQL | 872ms | 5 rows

Filter result... | Copy

Row	COVIDJSON
1	{ "Call To Action enabled": "FALSE", "Covid Banner": "FALSE", "Grubhub enabled": "FALSE", "Request a Quote Enabled": "FALSE", "Temporary Closed Until": "FALSE", "Virtual Services Offered": "FALSE", "business_id": "9kXRUlkwdDtnAPO6tVo51g", ...
2	{ "Call To Action enabled": "FALSE", "Covid Banner": "FALSE", "Grubhub enabled": "FALSE", "Request a Quote Enabled": "FALSE", "Temporary Closed Until": "FALSE", "Virtual Services Offered": "FALSE", "business_id": "H6D5HOTfMjrZt7r1EObZ1g", ...
3	{ "Call To Action enabled": "FALSE", "Covid Banner": "FALSE", "Grubhub enabled": "FALSE", "Request a Quote Enabled": "FALSE", "Temporary Closed Until": "FALSE", "Virtual Services Offered": "FALSE", "business_id": "FYddq7fUtzobZcw4jOJgVA", ...
4	{ "Call To Action enabled": "FALSE", "Covid Banner": "FALSE", "Grubhub enabled": "FALSE", "Request a Quote Enabled": "FALSE", "Temporary Closed Until": "FALSE", "Virtual Services Offered": "FALSE", "business_id": "c75jLTjIgA9q3glmLEGT6w", ...
5	{ "Call To Action enabled": "FALSE", "Covid Banner": "FALSE", "Grubhub enabled": "FALSE", "Request a Quote Enabled": "FALSE", "Temporary Closed Until": "FALSE", "Virtual Services Offered": "FALSE", "business_id": "YfzPiY50h_10SjIg3mnNWQ", ...

Figure 05 – Shows first 05 rows of the covid table created in the staging schema

3.5 CUSTOMER Table:

```
1 SELECT *
2 FROM UDACITYPROJECT.STAGING.CUSTOMER
3 LIMIT 5;
```

Results Data Preview

Query ID SQL 847ms 5 rows

Filter result... Copy

Row	CUSTOMERJSON
1	{ "average_stars": 3.74, "compliment_cool": 409, "compliment_cute": 4, "compliment_funny": 409, "compliment_hot": 226, "compliment_list": 3, "compliment_more": 18, "compliment_note": 116, "compliment_photos": 87, "compliment_plain": ...
2	{ "average_stars": 4.52, "compliment_cool": 0, "compliment_cute": 0, "compliment_funny": 0, "compliment_hot": 0, "compliment_list": 0, "compliment_more": 1, "compliment_note": 1, "compliment_photos": 0, "compliment_plain": 1, "comp...
3	{ "average_stars": 3.94, "compliment_cool": 11, "compliment_cute": 0, "compliment_funny": 11, "compliment_hot": 11, "compliment_list": 0, "compliment_more": 3, "compliment_note": 5, "compliment_photos": 2, "compliment_plain": 13, "...
4	{ "average_stars": 3.55, "compliment_cool": 8, "compliment_cute": 0, "compliment_funny": 8, "compliment_hot": 4, "compliment_list": 0, "compliment_more": 1, "compliment_note": 10, "compliment_photos": 5, "compliment_plain": 6, "com...
5	{ "average_stars": 4.31, "compliment_cool": 0, "compliment_cute": 0, "compliment_funny": 0, "compliment_hot": 0, "compliment_list": 0, "compliment_more": 0, "compliment_note": 3, "compliment_photos": 0, "compliment_plain": 1, "comp...

Figure 06 – Shows first 05 rows of the covid table created in the staging schema

3.6 TIP Table:

```
1 SELECT *
2 FROM UDACITYPROJECT.STAGING.TIP
3 LIMIT 5;
```

Results Data Preview

Query ID SQL 1.11s 5 rows

Filter result... Copy

Row	TIPJSON
1	{ "business_id": "7kHvPCBSmRgzDgjO9xXT1g", "compliment_count": 0, "date": "2012-03-22 22:29:37", "text": "Friendly service!", "user_id": "1f5cUfo7Il8D0_UDyyMgkQ" }
2	{ "business_id": "qck6waU6WlXBesRKmtsWfg", "compliment_count": 0, "date": "2011-12-27 20:54:57", "text": "Loves me some Kaldi's Coffee. Best this side of Columbia.", "user_id": "kLpFct8NWGTsPa3kS48A9g" }
3	{ "business_id": "ompDR5sUDpoI6gnTIdmneQ", "compliment_count": 0, "date": "2017-06-15 22:04:00", "text": "The food was very good.", "user_id": "_TtOBGYhWVm0IO6s6ZEI0g" }
4	{ "business_id": "pgV1yPVZ7TKmS3lvbCffNQ", "compliment_count": 0, "date": "2013-05-31 19:48:58", "text": "This place rocks, even if you're not into fishing hunting or all the sports offered.... It's cool to walk around and look at the displays",...
5	{ "business_id": "ZpQs6ICV63NwLrTO9kNHAA", "compliment_count": 0, "date": "2012-01-12 01:54:08", "text": "I just", "user_id": "08e8D5VZtixje3KVZkcJPw" }

Figure 07 – Shows first 05 rows of the tip table created in the staging schema

3.7 REVIEW Table:

```
1 SELECT *
2 FROM UDACITYPROJECT.STAGING.REVIEW
3 LIMIT 5;
```

Results Data Preview

Query ID SQL 796ms 5 rows

Filter result... Copy

Row	REVIEWJSON
1	{ "business_id": "X9e-ITfOeY0z6C-VAiSf5Q", "cool": 0, "date": "2020-09-18 05:48:51", "funny": 0, "review_id": "msaag1iuBorxy4DKIGSujg", "stars": 1, "text": "This business is doing everything in its power to shut me up and I REFUSE TO ALLOW...
2	{ "business_id": "JvawJ9bSr22xn4R9oLvl_w", "cool": 0, "date": "2021-04-14 23:51:10", "funny": 0, "review_id": "9R44VA73y_JOQgYhifJuLg", "stars": 5, "text": "Acacia was amazing! The oysters were amazing as well! We will definitely be back!...
3	{ "business_id": "EtKSTHV5Qx_Q7Aur9o4kQQ", "cool": 1, "date": "2018-10-12 01:01:17", "funny": 1, "review_id": "wrOGPch2IZZ0xc0nJfzt8Q", "stars": 3, "text": "***Update****\n\nI have updated my score as the general manager at Village Whiskey...
4	{ "business_id": "VQC5whJNU6Nia95jpz927A", "cool": 0, "date": "2013-02-26 16:33:57", "funny": 0, "review_id": "kthkK_rtnNnaQID-9KGfFQ", "stars": 5, "text": "I had a 50\" rear projection tv in my basement that weighed 250~lbs and it was NOT...
5	{ "business_id": "zNua_GyqRLb7AA7C6D9LmA", "cool": 0, "date": "2021-04-18 18:03:28", "funny": 0, "review_id": "I17EjxRqr9c_95XepwcAUQ", "stars": 5, "text": "Best hot yoga studio in the area! All of the instructors are amazing - Sara will p...

Figure 08 – Shows first 05 rows of the review table created in the staging schema

Screenshot of 2 tables created upon upload of climate data

Two climates data files with file size smaller than 50 MB have been uploaded into Snowflake using browser.

3.8 TEMPERATURE TABLE

```
1 SELECT *
2 FROM UDACITYPROJECT.STAGING.TEMPERATURE
3 LIMIT 5;
```

Results Data Preview

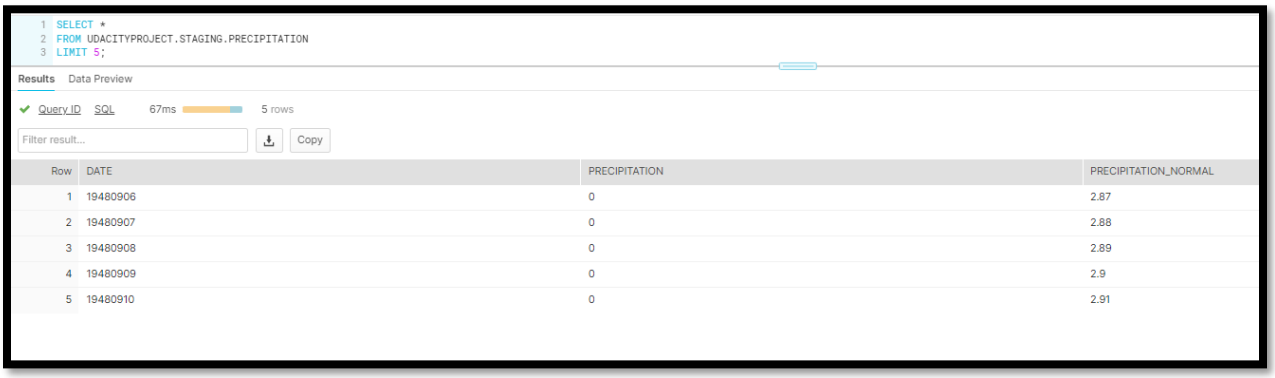
Query ID SQL 474ms 5 rows

Filter result... Copy

Row	DATE	MIN_TEMP	MAX_TEMP	NORMAL_MIN	NORMAL_MAX
1	19480906	105	61	97.7	74.9
2	19480907	104	60	97.4	74.5
3	19480908	103	64	97.1	74.2
4	19480909	106	62	96.7	73.8
5	19480910	106	61	96.4	73.5

Figure 09 – Shows first 05 rows of the temperature table created in the staging schema

3.9 PRECIPITATION Table:



```
SELECT *
FROM UDACITYPROJECT.STAGING.PRECIPITATION
LIMIT 5;
```

Results Data Preview

Query ID SQL 67ms 5 rows

Filter result... Copy

Row	DATE	PRECIPITATION	PRECIPITATION_NORMAL
1	19480906	0	2.87
2	19480907	0	2.88
3	19480908	0	2.89
4	19480909	0	2.9
5	19480910	0	2.91

Figure 10 – Shows first 05 rows of the precipitation table created in the staging schema

4. ODS

4.1 Transforming Staging to ODS

SQL code that transforms staging to ODS.

```
-- CREATE TEMPERATURE TABLE IN ODS SCHEMA
CREATE OR REPLACE TABLE TEMPERATURE (

                        DATE          DATE PRIMARY KEY,
                        MIN_TEMP      NUMBER,
                        MAX_TEMP      NUMBER,
                        NORMAL_MIN    FLOAT,
                        NORMAL_MAX    FLOAT

);

-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.TEMPERATURE TO
ODS.TEMPERATURE TABLE
insert into TEMPERATURE(DATE,
                      MIN_TEMP,
                      MAX_TEMP,
                      NORMAL_MIN,
                      NORMAL_MAX)

SELECT
        TO_DATE(DATE,'YYYYMMDD'),
        MIN_TEMP::NUMBER,
```

```
        MAX_TEMP::NUMBER,
        NORMAL_MIN::FLOAT,
        NORMAL_MAX::FLOAT
FROM UDACITYPROJECT.STAGING.TEMPERATURE;


-- CREATE PRECIPITATION TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE PRECIPITATION (

                          DATE          DATE PRIMARY KEY,
                          PRECIPITATION VARCHAR,
                          PRECIPITATION_NORMAL INTEGER
);

-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.PRECIPITATION TO
ODS.PRECIPITATION TABLE

INSERT INTO PRECIPITATION  (DATE,
                            PRECIPITATION,
                            PRECIPITATION_NORMAL )

SELECT TO_DATE(DATE,'YYYYMMDD'),
       PRECIPITATION::VARCHAR,
       PRECIPITATION_NORMAL::INTEGER
FROM UDACITYPROJECT.STAGING.PRECIPITATION;

-- CREATE BUSINESS TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE BUSINESS (

                            BUSINESS_ID   VARCHAR PRIMARY KEY,
                            ADDRESS       VARCHAR,
                            ATTRIBUTES    OBJECT,
                            CATEGORIES    VARCHAR,
                            HOURS         VARCHAR,
                            IS_OPEN       INTEGER,
                            LATITUDE      FLOAT,
                            LONGITUDE     FLOAT,
                            NAME          VARCHAR,
                            POSTAL_CODE   VARCHAR,
                            REVIEW_COUNT  INTEGER,
                            STARS         INTEGER,
                            STATE         VARCHAR);
```

```sql
-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.BUSINESS  TO ODS.BUSINESS
TABLE

INSERT INTO BUSINESS

    select  businessjson:business_id::varchar,
            businessjson:address::varchar,
            businessjson:attributes::object,
            businessjson:categories::varchar,
            businessjson:hours::varchar,
            businessjson:is_open::integer,
            businessjson:lattitude::float,
            businessjson:longitude::float,
            businessjson:name::string,
            businessjson:postal_code::varchar,
            businessjson:review_count::integer,
            businessjson:stars::integer,
            businessjson:state::varchar

FROM UDACITYPROJECT.STAGING.BUSINESS;



-- CREATE CUSTOMER TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE CUSTOMER (

                            USER_ID                 VARCHAR PRIMARY KEY,
                            NAME                    VARCHAR,
                            YELPING_SINCE           DATE,
                            AVERAGE_STARS           FLOAT,
                            COMPLIMENT_COOL         INTEGER,
                            COMPLIMENT_CUTE         INTEGER,
                            COMPLIMENT_FUNNY        INTEGER,
                            COMPLIMENT_HOT          INTEGER,
                            COMPLIMENT_LIST         INTEGER,
                            COMPLIMENT_MORE         INTEGER,
                            COMPLIMENT_NOTE         INTEGER,
                            COMPLIMENT_PHOTOS       INTEGER,
                            COMPLIMENT_PLAIN        INTEGER,
                            COMPLIMENT_PROFILE      INTEGER,
                            COMPLIMENT_WRITER       INTEGER,
                            COOL                    INTEGER,
                            ELITE                   VARCHAR,
                            FANS                    INTEGER,
```

```
                            FRIENDS                 VARCHAR,
                            FUNNY                   INTEGER,
                            REVIEW_COUNT            INTEGER,
                            USEFUL                  INTEGER

);



-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.CUSTOMER TO ODS.CUSTOMER
INSERT INTO CUSTOMER
    select
            customerjson:user_id::varchar,
            customerjson:name::varchar,
            to_date(customerjson:yelping_since::string),
            customerjson:average_stars::varchar,
            customerjson:compliment_cool::integer,
            customerjson:compliment_cute::integer,
            customerjson:compliment_funny::integer,
            customerjson:compliment_hot::integer,
            customerjson:compliment_list::integer,
            customerjson:compliment_more::integer,
            customerjson:compliment_note::integer,
            customerjson:compliment_photos::integer,
            customerjson:compliment_plain::integer,
            customerjson:compliment_profile::integer,
            customerjson:compliment_writer::integer,
            customerjson:cool::integer,
            customerjson:elite::varchar,
            customerjson:fans::integer,
            customerjson:friends::VARCHAR,
            customerjson:funny::integer,

            customerjson:review_count::integer,
            customerjson:useful::integer



FROM UDACITYPROJECT.STAGING.CUSTOMER;
```

```sql
-- CREATE TIP TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE TIP (

                            TIP_ID              NUMBER AUTOINCREMENT PRIMARY KEY,
                            BUSINESS_ID         VARCHAR,
                            COMPLIMENT_COUNT    INTEGER,
                            DATE                DATE,
                            TEXT                VARCHAR,
                            USER_ID             VARCHAR

);

-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.TIP  TO ODS.TIP
INSERT INTO TIP  (          BUSINESS_ID,
                            COMPLIMENT_COUNT,
                            DATE,
                            TEXT,
                            USER_ID)

    select
            tipjson:business_id::varchar,
            tipjson:compliment_count::integer,
            to_date(tipjson:date::string),
            tipjson:text::varchar,
            tipjson:user_id::varchar


FROM UDACITYPROJECT.STAGING.TIP;


-- CREATE REVIEW TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE  REVIEW (

                            REVIEW_ID     VARCHAR PRIMARY KEY,
                            BUSINESS_ID   VARCHAR,
                            COOL          INTEGER,
                            DATE          DATE,
                            FUNNY         INTEGER,
                            STARS         INTEGER,
                            TEXT          VARCHAR,
                            USEFUL        INTEGER,
                            USER_ID       VARCHAR
);
```

```sql
-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.REVIEW  TO ODS.REVIEW
INSERT INTO REVIEW

    select
            reviewjson:review_id::varchar,
            reviewjson:business_id::varchar,
            reviewjson:cool::integer,
            to_date(reviewjson:date::string),
            reviewjson:funny::integer,
            reviewjson:stars::integer,
            reviewjson:text::varchar,
            reviewjson:useful::integer,
            reviewjson:user_id::varchar


FROM UDACITYPROJECT.STAGING.REVIEW;



CREATE OR REPLACE TABLE  COVID (

                            ID                          NUMBER AUTOINCREMENT
PRIMARY KEY,
                            BUSINESS_ID                 VARCHAR,
                            CALL_TO_ACTION_ENABLED      VARCHAR,
                            COVID_BANNER                VARCHAR,
                            GRUBHUB_ENABLED             VARCHAR,
                            REQUEST_A_QUOTE_ENABLED     VARCHAR,
                            TEMPORARY_CLOSED_UNTIL      VARCHAR,
                            VIRTUAL_SERVICES_OFFERED    VARCHAR,
                            DELIVERY_OR_TAKEOUT         VARCHAR,
                            HIGHLIGHTS                  VARCHAR

);
```

```sql
INSERT INTO COVID (
                            BUSINESS_ID,
                            CALL_TO_ACTION_ENABLED,
                            COVID_BANNER,
                            GRUBHUB_ENABLED,
                            REQUEST_A_QUOTE_ENABLED,
                            TEMPORARY_CLOSED_UNTIL,
                            VIRTUAL_SERVICES_OFFERED,
                            DELIVERY_OR_TAKEOUT,
                            HIGHLIGHTS)

select
            covidjson:business_id::VARCHAR,
            covidjson:"Call To Action enabled"::VARCHAR,
            covidjson:"Covid Banner"::VARCHAR,
            covidjson:"Grubhub enabled"::VARCHAR,
            covidjson:"Request a Quote Enabled"::VARCHAR,
            covidjson:"Temporary Closed Until"::VARCHAR,
            covidjson:"Virtual Services Offered"::VARCHAR,
            covidjson:"delivery or takeout"::VARCHAR,
            covidjson:highlights::VARCHAR

FROM UDACITYPROJECT.STAGING.COVID;




-- CREATE CHECKIN TABLE IN ODS SCHEMA

CREATE OR REPLACE TABLE  CHECKIN (

                            BUSINESS_ID   VARCHAR PRIMARY KEY,
                            DATE          VARCHAR);


-- COPY DATA FROM STAGING SCHEMA TO ODS FROM STAGING.REVIEW  TO ODS.REVIEW
INSERT INTO CHECKIN

    select
            checkinjson:business_id::varchar,
            checkinjson::varchar

FROM UDACITYPROJECT.STAGING.CHECKIN;
```

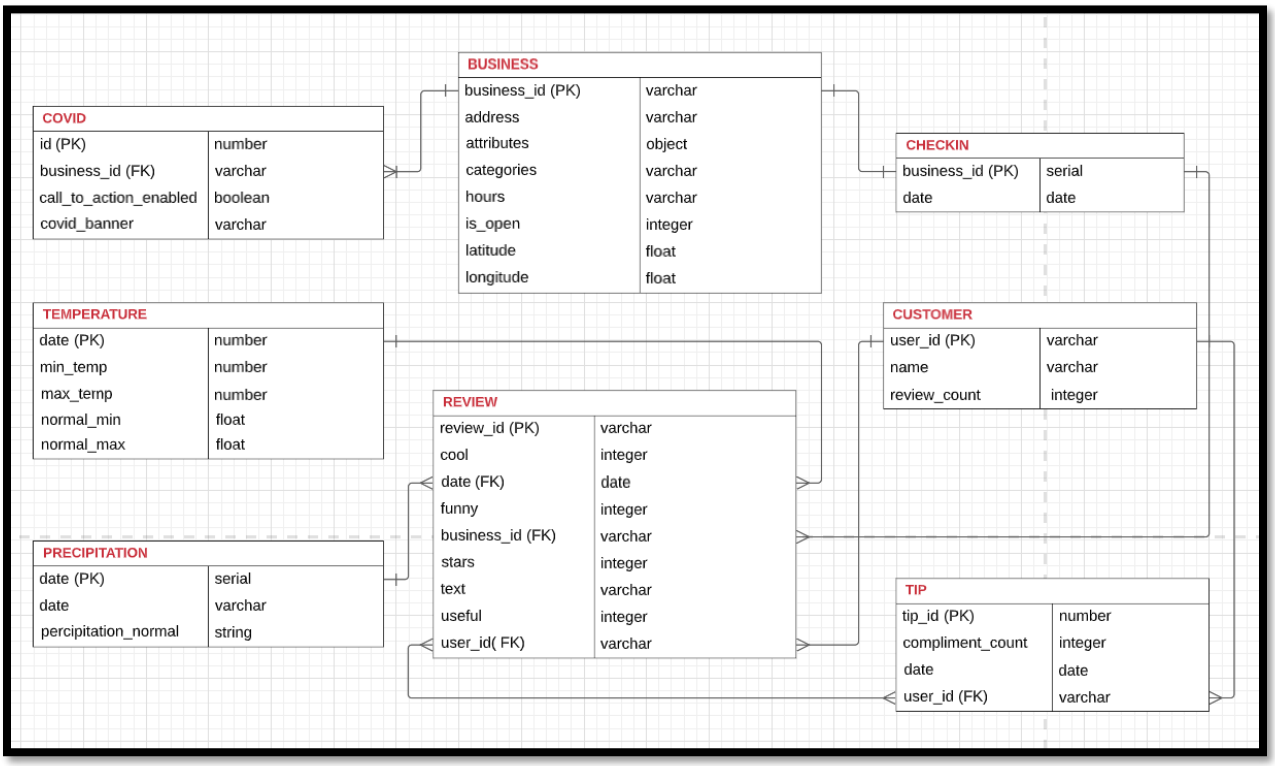
Screenshot that communicates different sizes/row_counts of raw, staging, and ODS tables in database:

Table -1 : Summarizes the size of the raw files, staging, and ODS.

File Name	Raw file size	Staging file size	ODS file size
Temperature	709 KB	195 KB	172.5 KB
Precipitation	536 KB	126.5 KB	85KB
Business	116.078 MB	11.0 MB	9.9MB
Check-in	280.234 MB	80.5 MB	86.4MB
Covid	63.316 MB	5.0 MB	5.0MB
Review	5.216669 GB	1.9 GB	1.9GB
Tip	176.372 MB	46.1 MB	40.5MB
Customer	3.28 GB	1.8 GB	1.8GB

4.2 ER diagram ODS SCHEMAS

Figure 11 – Shows ER diagram among the ODS tables



- SQL query code evidence that shows how the datasets are integrated

```sql
SELECT   b.business_id,
         b.attributes,
         r.date,
         r.stars,
         t.max_temp,
         p.precipitation,
         c.user_id,
         co.call_to_action_enabled,
         tip.compliment_count



  FROM review r

  JOIN precipitation p
  ON r.date = p.date

  JOIN temperature t
  ON r.date = t.date

  JOIN checkin ch
  ON r.business_id = ch.business_id

  JOIN business b
  ON b.business_id = ch.business_id

  JOIN covid co
  ON co.business_id = b.business_id

  JOIN customer c
  ON r.user_id = c.user_id

  JOIN tip
  ON tip.user_id = r.user_id

  LIMIT 5;
```

Filter result... Copy Columns

Row	BUSINESS_ID	ATTRIBUTES	DATE	STARS	MAX_TEMP	PRECIPITATION	USER_ID	CALL_TO_ACTION_ENAI	COMPLIMENT_COUNT
1	gPxoK-31HK-IRc4WAX7kxw	{ "BikeParking": "False", "Busin...	2014-02-28	5	48	0.3	iimX6RwTcZ...	FALSE	0
2	gPxoK-31HK-IRc4WAX7kxw	{ "BikeParking": "False", "Busin...	2014-02-28	5	48	0.3	iimX6RwTcZ...	FALSE	0
3	gPxoK-31HK-IRc4WAX7kxw	{ "BikeParking": "False", "Busin...	2014-02-28	5	48	0.3	iimX6RwTcZ...	FALSE	0
4	gPxoK-31HK-IRc4WAX7kxw	{ "BikeParking": "False", "Busin...	2014-02-28	5	48	0.3	iimX6RwTcZ...	FALSE	0
5	gPxoK-31HK-IRc4WAX7kxw	{ "BikeParking": "False", "Busin...	2014-02-28	5	48	0.3	iimX6RwTcZ...	FALSE	0

Figure 12– Table showing column both from temperature and YELP data that shows that the data are related.

5. DWH SCEMA

5.1 Diagram of star schema

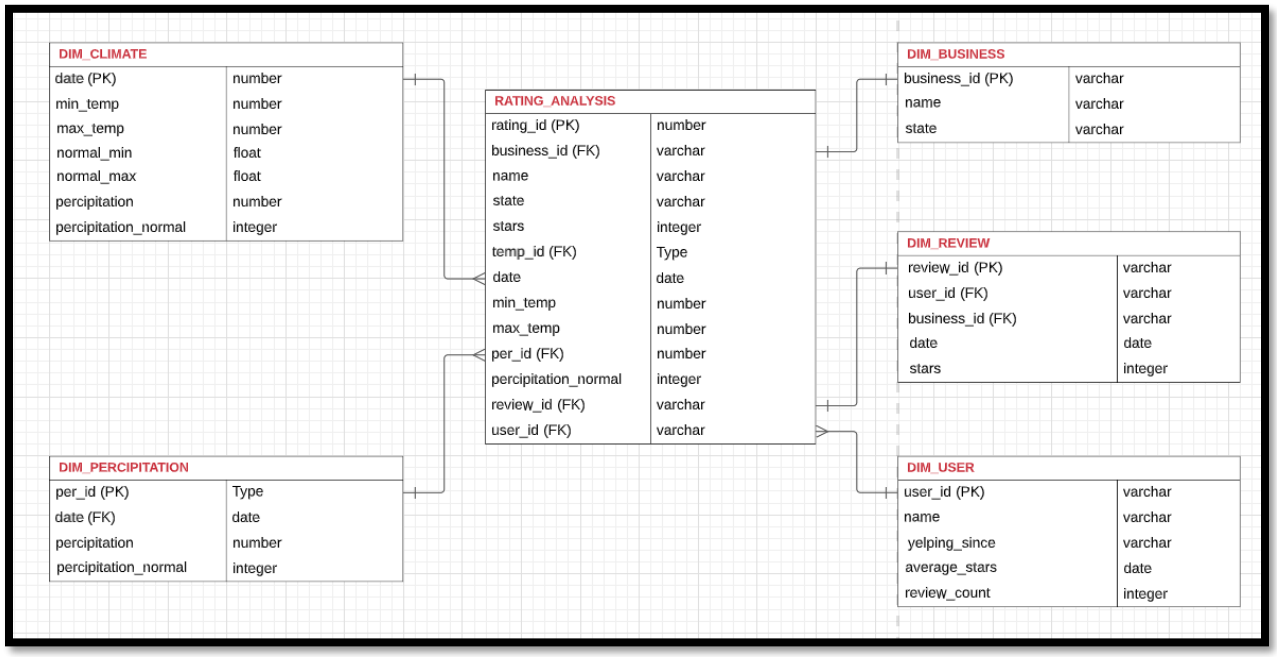


Figure 13– Shows STAR schema made for the analytical reporting purpose

A Fact can be described as- "a collection of related data items, which consist of measures and context data. Each fact typically represents a business item, a business transaction, or an event that can be used in analyzing the business or business processes". In this case a business transaction is the rating of a particular restaurant at any given date and we are interested in analyzing the impact of weather (i.e temperature and precipitation) on restaurant rating.

Hence, the Fact Table contains records associated with restaurant rating such as business_id, name, state, stars, date, min_temp, max_temp, precipitation_normal etc.

Dimension Table: The dimension table contains the attributes related to a restaurent and help to store the context of the business. The dimension table contains the following items-

- **DIM_REVIEW:** contains the rating by an user at a given instance and have the following columns- review_id, business_id, date, stars
- **DIM_BUSINESS:** contains the information, name and location of a restaurant - business_id, name, and state
- **DIM_TEMPERATURE:** contains the information related to temperature. It has the properties of date and temperatures.
- **DIM_PERCIPITATION:** contains the information related to participation at any given day.

- SQL queries code necessary to move the data from ODS to DWH.

```
-- CREATE BUSINESS TABLE IN WAREHOUSE SCHEMA

CREATE OR REPLACE TABLE DIM_BUSINESS (

                          BUSINESS_ID   VARCHAR PRIMARY KEY,
                          NAME          VARCHAR,
                          STATE         VARCHAR);


INSERT INTO DIM_BUSINESS

SELECT DISTINCT BUSINESS_ID,
                NAME,
                STATE

FROM ODS.BUSINESS;
```

```sql
-- CREATE REVIEW TABLE IN WAREHOUSE SCHEMA
CREATE OR REPLACE TABLE  DIM_REVIEW (

                            REVIEW_ID     VARCHAR PRIMARY KEY,
                            USER_ID       VARCHAR,
                            BUSINESS_ID   VARCHAR,
                            DATE          DATE,
                            STARS         INTEGER,

constraint fk_BUSINESS_ID foreign key (BUSINESS_ID)
references ODS.BUSINESS (BUSINESS_ID),


constraint fk_USER_ID foreign key (USER_ID)
references ODS.CUSTOMER(USER_ID));

INSERT INTO DIM_REVIEW

SELECT          REVIEW_ID,
                USER_ID,
                BUSINESS_ID,
                DATE,
                STARS

FROM ODS.REVIEW;

-- CREATE CUSTOMER TABLE IN WAREHOUSE SCHEMA
CREATE OR REPLACE TABLE  DIM_USER (

                            USER_ID                 VARCHAR PRIMARY KEY,
                            NAME                    VARCHAR,
                            YELPING_SINCE           DATE,
                            AVERAGE_STARS           FLOAT,
                            REVIEW_COUNT            INTEGER);
INSERT INTO DIM_USER

SELECT
                USER_ID,
                NAME,
                YELPING_SINCE,
                AVERAGE_STARS,
                REVIEW_COUNT


FROM ODS.CUSTOMER;
```

```sql
-- CREATE TEMPERATURE TABLE IN WAREHOUSE SCHEMA
CREATE OR REPLACE TABLE DIM_CLIMATE (

                        DATE                DATE PRIMARY KEY,
                        MIN_TEMP            NUMBER,
                        MAX_TEMP            NUMBER,
                        NORMAL_MIN          FLOAT,
                        NORMAL_MAX          FLOAT,
                        PRECIPITATION       VARCHAR,
                        PRECIPITATION_NORMAL INTEGER);


INSERT INTO DIM_CLIMATE

select distinct
               T.DATE,
               MIN_TEMP,
               MAX_TEMP,
               NORMAL_MIN,
               NORMAL_MAX,
               P.PRECIPITATION,
               P.PRECIPITATION_NORMAL


FROM ODS.TEMPERATURE T
JOIN ODS.PRECIPITATION p
ON T.DATE = P.DATE;
```

- SQL queries code that reports the business name, temperature, precipitation, and ratings.

  Create table rating analysis that can be used for reporting purposes:

```sql
CREATE OR REPLACE TABLE RATING_ANALYSIS (

                          RATING_ID           NUMBER AUTOINCREMENT,
                          BUSINESS_ID         VARCHAR,
                          NAME                VARCHAR,
                          STATE               VARCHAR,
                          STARS               INTEGER,
```

```
                            TEMP_ID              NUMBER,
                            DATE                 DATE,
                            MIN_TEMP             NUMBER,
                            MAX_TEMP             NUMBER,
                            PER_ID               NUMBER,
                            PRECIPITATION_NORMAL INTEGER,
                            REVIEW_ID            VARCHAR,

constraint fk_BUSINESS_ID foreign key (BUSINESS_ID)
references ODS.BUSINESS (BUSINESS_ID),

constraint fk_REVIEW_ID foreign key (REVIEW_ID)
references ODS.REVIEW (REVIEW_ID),


constraint fk_DATE_TEMP foreign key (TEMP_ID)
references ODS.TEMPERATURE(TEMP_ID),


constraint fk_DATE_PERCIPTATION foreign key (PER_ID)
references ODS.PRECIPITATION(PER_ID)

);
```

Insert data into the table:

```
CREATE OR REPLACE TABLE RATING_ANALYSIS (

                            RATING_ID            NUMBER AUTOINCREMENT,
                            BUSINESS_ID          VARCHAR,
                            NAME                 VARCHAR,
                            STATE                VARCHAR,
                            STARS                INTEGER,
                            DATE                 DATE,
                            MIN_TEMP             NUMBER,
                            MAX_TEMP             NUMBER,
                            PRECIPITATION_NORMAL INTEGER,
                            USER_ID              VARCHAR,
                            AVERAGE_STARS        FLOAT,
                            REVIEW_ID            VARCHAR,


constraint fk_BUSINESS_ID foreign key (BUSINESS_ID)
references ODS.BUSINESS (BUSINESS_ID),

constraint fk_REVIEW_ID foreign key (REVIEW_ID)
references ODS.REVIEW (REVIEW_ID),


constraint fk_DATE_TEMP foreign key (DATE)
references ODS.TEMPERATURE(DATE),


constraint fk_DATE_PERCIPTATION foreign key (DATE)
references ODS.PRECIPITATION(DATE),

constraint fk_USER_ID foreign key (USER_ID)
references ODS.CUSTOMER(USER_ID)

);

INSERT INTO RATING_ANALYSIS ( BUSINESS_ID,
                              NAME,
                              STATE,
                              STARS,
                              DATE,
                              MIN_TEMP,
                              MAX_TEMP,
```

```
                                PRECIPITATION_NORMAL,
                                USER_ID,
                                AVERAGE_STARS,
                                REVIEW_ID)

SELECT          B.BUSINESS_ID,
                B.NAME,
                B.STATE,
                R.STARS,
                T.DATE,
                T.MIN_TEMP,
                T.MAX_TEMP,
                P.PRECIPITATION_NORMAL,
                C.USER_ID,
                C.AVERAGE_STARS,
                R.REVIEW_ID




FROM ODS.BUSINESS B,
     ODS.TEMPERATURE T,
     ODS.PRECIPITATION P,
     ODS.REVIEW R,
     ODS.CUSTOMER C

WHERE (T.DATE = P.DATE)
      AND (T.DATE = R.DATE)
      AND R.BUSINESS_ID =  B.BUSINESS_ID
      AND C.USER_ID = R.USER_ID;
```

Row	RATING_ID	BUSINESS_ID	NAME	STATE	STARS	DATE	MIN_TEMP	MAX_TEMP	PRECIPITATION_N(	USER_ID	AVERAGE_STARS	REVIEW_ID
1	6064562	XTsiutArkvTDr-...	Bulk Nation	FL	4	2017-03-20	88	68	2	4nrrSY5K4n4o...	3.46	EYP49OKvdfkc...
2	6064563	mKqlE-HZViEz...	Loews Ventana...	AZ	3	2018-03-27	70	51	2	n0_exasoAxiFRI...	2.15	A0Fm2Cg8yyZj...
3	6064564	YcblCqp_TlaxN...	Hickory Lane A...	PA	3	2018-06-24	104	84	2	e4hd3J5hDY9...	3.9	lxNJQ0uldLfZw...
4	6064565	X7yGrz7oDbt8...	Everest Cafe & ...	MO	4	2012-12-17	57	43	4	PYTq8KWW0U...	3.83	cwmjAE0DSrz3...
5	6064566	c-y_ORtoUTFD...	Heritage	NV	3	2014-07-06	104	81	2	yZigt9aRAja7x-...	3.47	sdoygvbhH2M...

Figure 13– A view of the SQL generated report that clearly includes business name, temperature, precipitation, and ratings.

6. APPENDIX- 1

A. Code for uploading large data file using CLI

```
-- SELECT DATABASE
USE DATABASE UDACITYPROJECT;

-- SELECT WAREHOUSE
USE WAREHOUSE COMPUTE_WH;

-- SELECT SCHEMA
USE SCHEMA STAGING;

-- CREATE JSON FILE FORMAT
create or replace file format myjsonformat type='JSON' strip_outer_array=true;

-- CREATE STAGING AREA
create or replace stage my_json_stage file_format = myjsonformat;


-- CREATE A TABLE FOR THE BUSINESS JSON FILE
create or replace table business(businessjson variant);


-- INSERT THE business.JSON file
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\business.json @my_json_stage auto_compress=true;


-- COPY business.JOSN file from staging area into table
copy into business from @my_json_stage/business.json.gz file_format=myjsonformat
on_error='skip_file';



-- CREATE A TABLE FOR THE COVID JSON FILE
create table covid(covidjson variant);

-- INSERT THE covid.JSON file
```

```
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\covid.json @my_json_stage auto_compress=true parallel=20;


-- COPY covid.JOSN file from staging area into table
copy into covid from @my_json_stage/covid.json.gz file_format=myjsonformat
on_error='skip_file';



-- CREATE A TABLE FOR THE CHECK_IN JSON FILE
create table checkin(checkinjson variant);

-- INSERT THE checkin.JSON file
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\checkin.json @my_json_stage auto_compress=true parallel=25;


-- COPY checkin.JSON file from staging area into table
copy into checkin from @my_json_stage/checkin.json.gz file_format=myjsonformat
on_error='skip_file';




-- CREATE A TABLE FOR THE REVIEW JSON FILE
create table review(reviewjson variant);

-- INSERT THE review.JSON file
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\review.json @my_json_stage auto_compress=true parallel=25;


-- COPY review.JSON  file from staging area into table
copy into review from @my_json_stage/review.json.gz file_format=myjsonformat
on_error='skip_file';



-- CREATE A TABLE FOR THE TIPS JSON FILE
create table tip(tipjson variant);

-- INSERT THE tip.JSON file
```

```
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\tip.json @my_json_stage auto_compress=true parallel=25;


-- COPY tip.JOSN file from staging area into table
copy into tip from @my_json_stage/tip.json.gz file_format=myjsonformat
on_error='skip_file';


-- CREATE A TABLE FOR THE CUSTOMER JSON FILE
create table customer(customerjson variant);

-- INSERT THE customer.JSON file
put
file:///C:\Users\jahid.razan\Desktop\Udacity_Data_viz\Data_Architect_Nanodegree\P
roject_02\user.json @my_json_stage auto_compress=true parallel=25;


-- COPY customer.JOSN file from staging area into table
copy into customer from @my_json_stage/user.json.gz file_format=myjsonformat
on_error='skip_file';
```

7. REFERENCE

- Deep Diving in the world of Data Warehousing. Link.