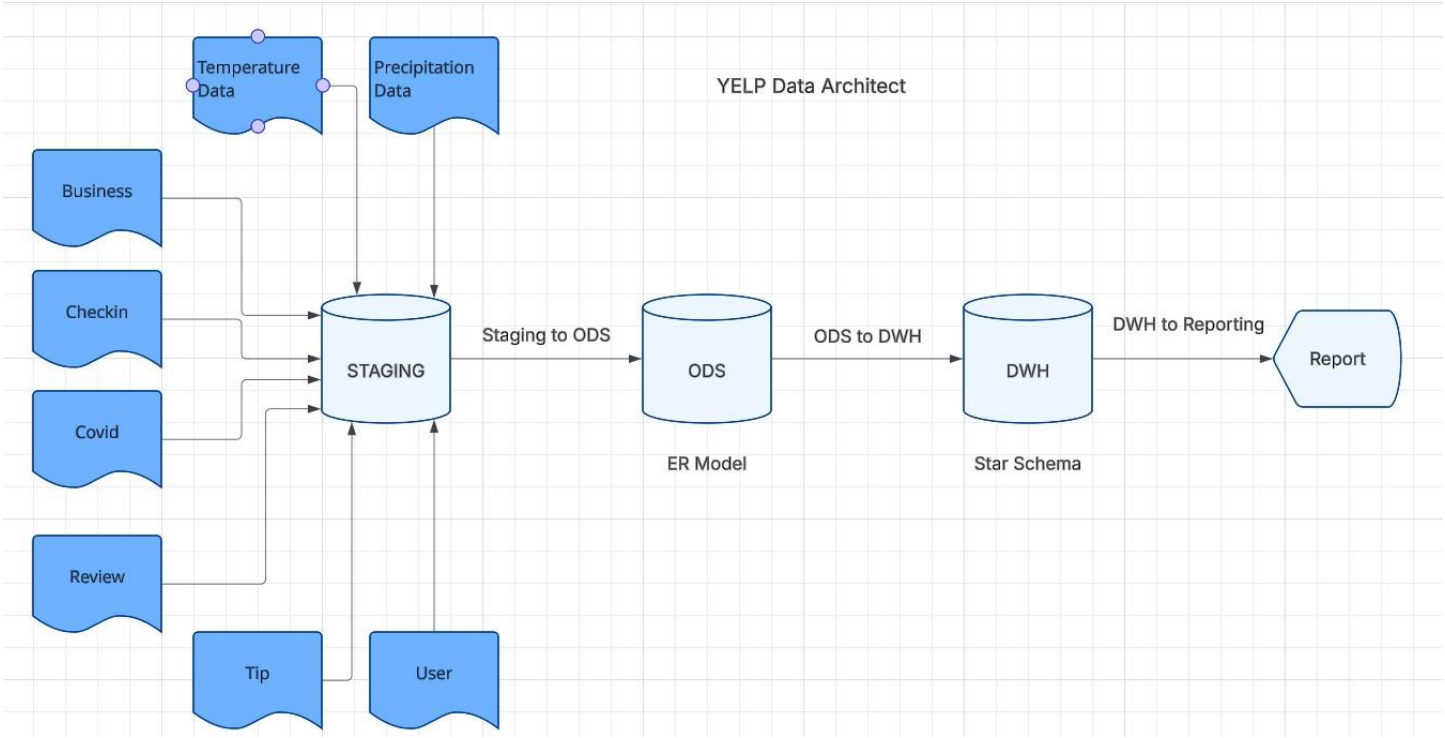
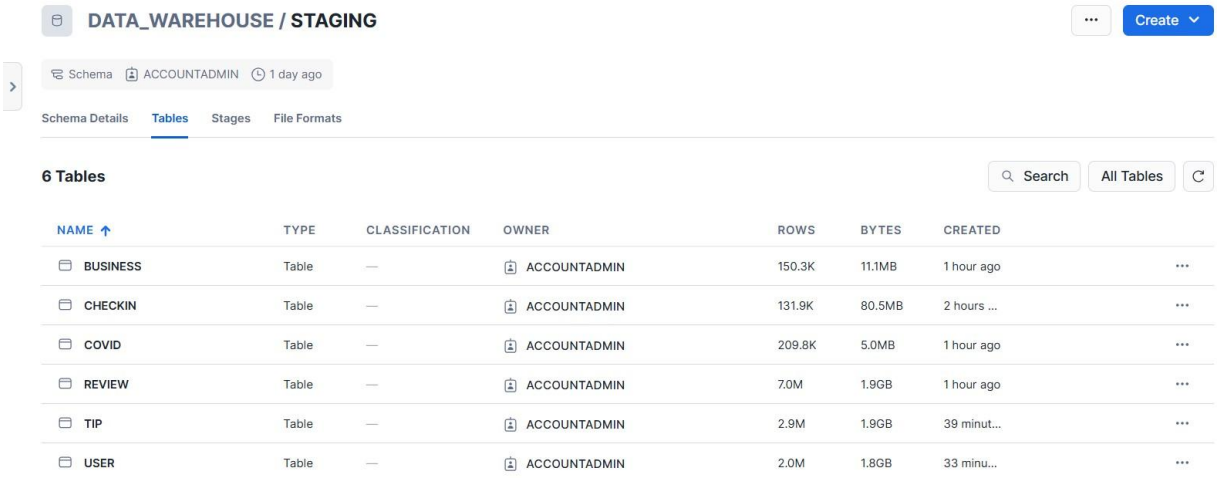
Submission Document - Design a Data Warehouse for Reporting and OLAP

A. Staging

1. Data architecture diagram showing all 8 files pointing to staging database to Operational Data Store (ODS) to Data Warehouse (DWH) to Reporting

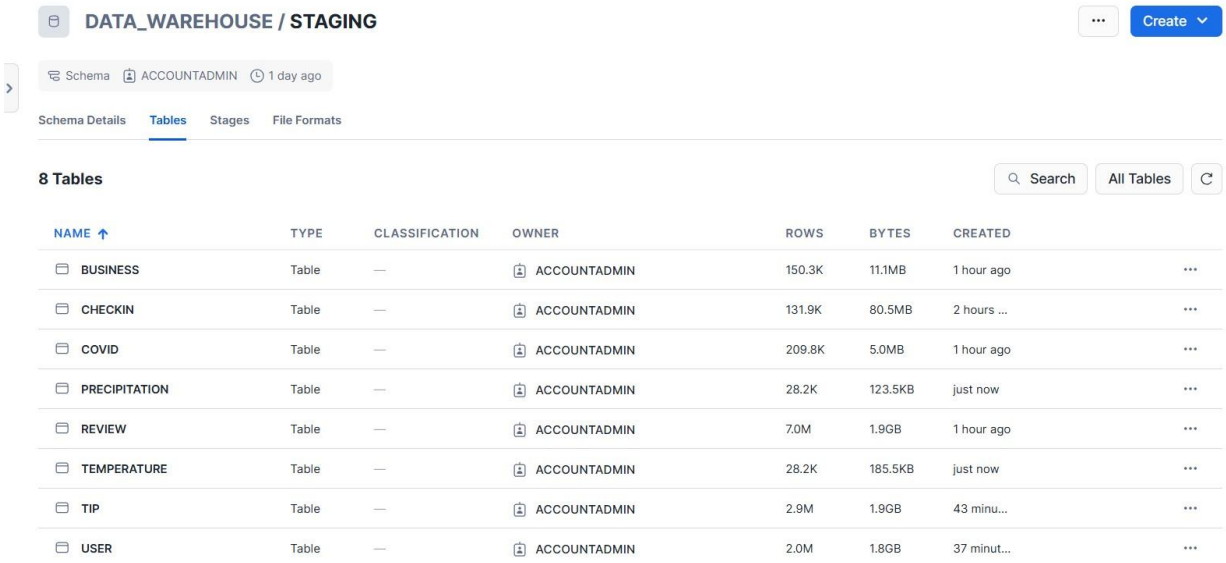


2. Screenshot showing the tables in the staging schema after extracting 6 Yelp files



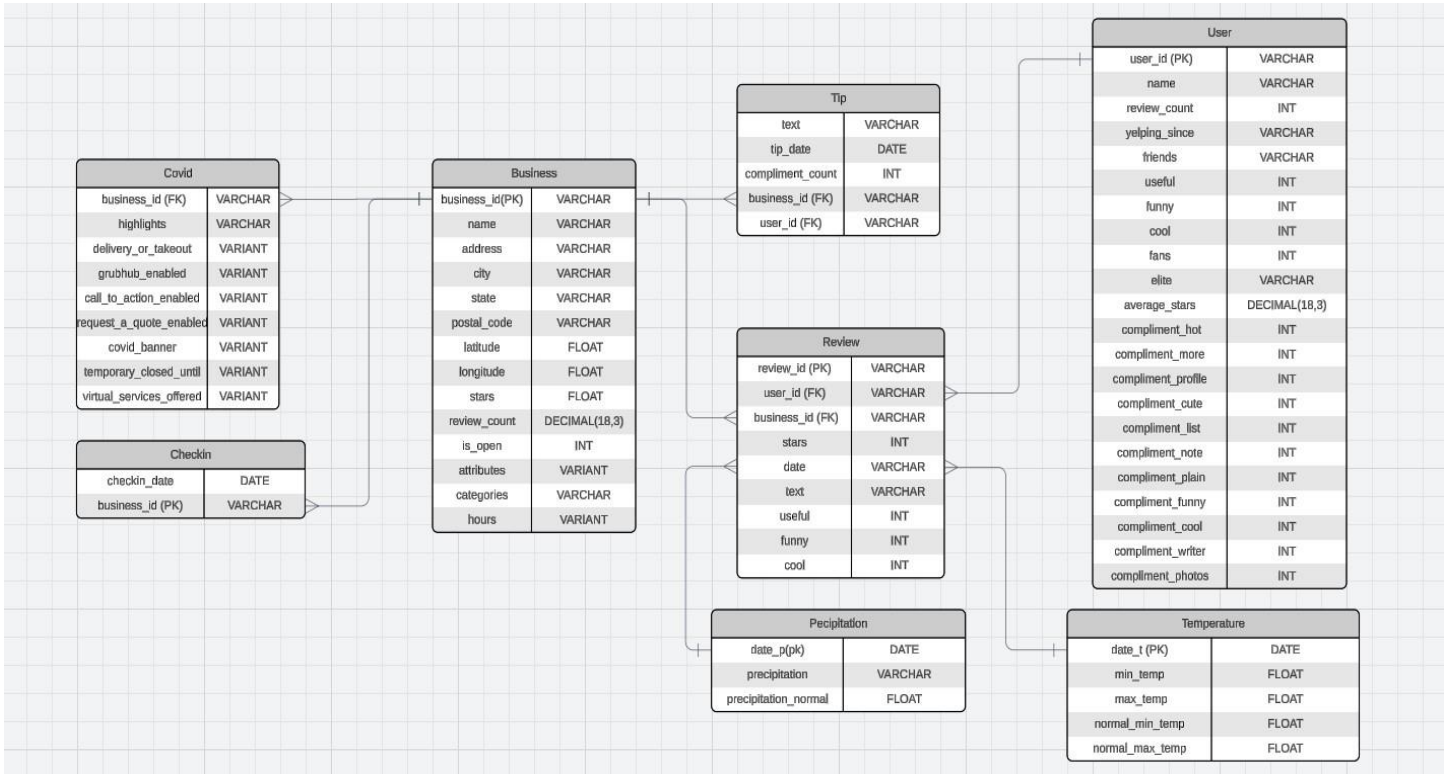
NAME	TYPE	CLASSIFICATION	OWNER	ROWS	BYTES	CREATED
BUSINESS	Table	—	ACCOUNTADMIN	150.3K	11.1MB	1 hour ago
CHECKIN	Table	—	ACCOUNTADMIN	131.9K	80.5MB	2 hours ...
COVID	Table	—	ACCOUNTADMIN	209.8K	5.0MB	1 hour ago
REVIEW	Table	—	ACCOUNTADMIN	7.0M	1.9GB	1 hour ago
TIP	Table	—	ACCOUNTADMIN	2.9M	1.9GB	39 minut...
USER	Table	—	ACCOUNTADMIN	2.0M	1.8GB	33 minu...

3. Screenshot showing the tables after extracting 2 files into the staging schema



B. Operational Data Store (ODS)

1. ER diagram that includes one-to-one and one-to-many relationships for tables: Business, Customer, Tips, Review, Precipitation, Covid, Check_in, Temperature



2. SQL queries that transform staging to ODS

```
--Business_d Table
CREATE TABLE business_D (
    business_id VARCHAR PRIMARY KEY,
    name VARCHAR ,
    address VARCHAR,
    city VARCHAR,
    state VARCHAR,
    postal_code VARCHAR,
    latitude FLOAT,
    longitude FLOAT,
    stars FLOAT,
    review_count DECIMAL(18,3),
    is_open INT,
    attributes VARIANT,
    categories VARCHAR,
    hours VARIANT
);
INSERT INTO business_D
SELECT
 usersjson:business_id,
    usersjson:name,
    usersjson:address,
    usersjson:city,
    usersjson:state,
    usersjson:postal_code,
    usersjson:latitude,
    usersjson:longitude,
    usersjson:stars,
    usersjson:review_count,
    usersjson:is_open,
    usersjson:attributes,
    usersjson:categories,
    usersjson:hours
FROM "DATA_WAREHOUSE"."STAGING".BUSINESS;

--user_d Table
CREATE TABLE user_D (
    user_id VARCHAR  PRIMARY KEY,
    name VARCHAR,
    review_count INT,
    yelping_since VARCHAR,
    friends VARCHAR,
    useful INT,
    funny INT,
```

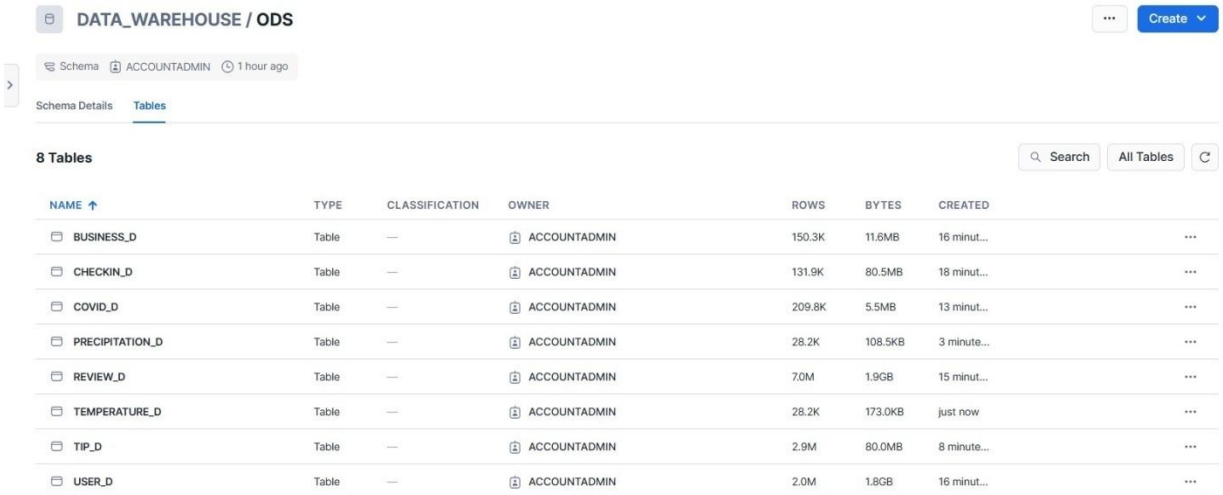
```
    cool INT,
    fans INT,
    elite VARCHAR,
    average_stars DECIMAL(18,3),
    compliment_hot INT,
    compliment_more INT,
    compliment_profile INT,
    compliment_cute INT,
    compliment_list INT,
    compliment_note INT,
    compliment_plain INT,
    compliment_funny INT,
    compliment_cool INT,
    compliment_writer INT,
    compliment_photos INT
);
INSERT INTO user_D
SELECT
    usersjson:user_id,
    usersjson:name,
    usersjson:review_count,
    usersjson:yelping_since,
    usersjson:friends,
    usersjson:useful,
    usersjson:funny,
    usersjson:cool,
    usersjson:fans,
    usersjson:elite,
    usersjson:average_stars,
    usersjson:compliment_hot,
    usersjson:compliment_more,
    usersjson:compliment_profile,
    usersjson:compliment_cute,
    usersjson:compliment_list,
    usersjson:compliment_note,
    usersjson:compliment_plain,
    usersjson:compliment_funny,
    usersjson:compliment_cool,
    usersjson:compliment_writer,
    usersjson:compliment_photos
FROM "DATA_WAREHOUSE"."STAGING".user;

--checkin_d Table
CREATE TABLE checkin_D (
    checkin_date VARCHAR,
    business_id VARCHAR,
    FOREIGN KEY (business_id) REFERENCES business_D(business_id)
);
INSERT INTO checkin_D
```

```
SELECT
    usersjson:date,
    usersjson:business_id
FROM "DATA_WAREHOUSE"."STAGING".checkin;

--review-d
CREATE TABLE review_D (
    review_id VARCHAR PRIMARY KEY,
    user_id VARCHAR,
    business_id VARCHAR,
    stars INT,
    date VARCHAR,
    text VARCHAR,
    useful INT,
    funny INT,
    cool INT
);
INSERT INTO review_D
SELECT
    usersjson:review_id,
    usersjson:user_id,
    usersjson:business_id,
    usersjson:stars,
    usersjson:date,
    usersjson:text,
    usersjson:useful,
    usersjson:funny,
    usersjson:cool
FROM "DATA_WAREHOUSE"."STAGING".review;

--covid_d Table
CREATE TABLE covid_D (
    business_id VARCHAR,
    highlights VARCHAR,
    delivery_or_takeout VARIANT,
    grubhub_enabled VARIANT,
    call_to_action_enabled VARIANT,
    request_a_quote_enabled VARIANT,
    covid_banner VARIANT,
    temporary_closed_until VARIANT,
    virtual_services_offered VARIANT,
    FOREIGN KEY (business_id) REFERENCES business_D(business_id)
);
INSERT INTO covid_D
SELECT
    usersjson:"business_id",
    usersjson:"highlights",
    usersjson:"delivery_or_takeout",
    usersjson:"grubhub_enabled",
```

```
    usersjson:"call_to_action_enabled",
    usersjson:"Request_a_quote_enabled",
    usersjson:"covid_banner",
    usersjson:"temporary_closed_until",
    usersjson:"virtual_services_offered"
FROM "DATA_WAREHOUSE"."STAGING".covid;

--tip_d Table
CREATE TABLE tip_D (
    text VARCHAR,
    tip_date VARCHAR,
    compliment_count INT,
    business_id VARCHAR,
    user_id VARCHAR,
    FOREIGN KEY (business_id) REFERENCES business_D(business_id),
    FOREIGN KEY (user_id) REFERENCES user_D(user_id)
);
INSERT INTO tip_D
SELECT
    usersjson:text,
    usersjson:tip_date,
    usersjson:compliment_count,
    usersjson:business_id,
    usersjson:user_id
FROM "DATA_WAREHOUSE"."STAGING".tip;

--precipitation_D Table
CREATE TABLE precipitation_D(
    date_p DATE PRIMARY KEY,
    precipitation VARCHAR,
    precipitation_normal FLOAT
 );
INSERT INTO precipitation_D
SELECT
    date("DATE",'YYYYMMDD'),
    precipitation,
    precipitation_normal
FROM "DATA_WAREHOUSE"."STAGING".precipitation;

--temperature_D
CREATE TABLE temperature_D(
 date_t DATE PRIMARY KEY,
 min_temp FLOAT,
 max_temp FLOAT,
 normal_min_temp FLOAT,
 normal_max_temp FLOAT
);
INSERT INTO temperature_D
SELECT
```

```
    date("DATE", 'YYYYMMDD'),
    min,
    max,
    normal_min,
    normal_max
FROM "DATA_WAREHOUSE"."STAGING".temperature;
```

3. Screenshot showing the queries were used successfully to transform the staging data to ODS



4. SQL queries that use JSON functions to transform staging data from a single JSON structure into multiple columns for ODS

-- JSON functions to transform staging data from a single JSON structure into multiple columns for ODS

```
----------------------------------------------------------------------------------------------------
-- Since we will be uploading a JSON file, we need to create a JSON file format.
create or replace file format myjsonformat type = "JSON" strip_outer_array = true;

--Next step is to create the Staging Area, which is a temporary holding area for data.
create or replace stage my_json_stage file_format = myjsonformat;

--Create a table with one column of type variant.
--business
create table business(usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_business.json @my_json_stage
auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_business.json.gz file_format=myjsonformat
on_error='skip_file';
```

--user
create table user (usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_user.json @my_json_stage auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_user.json.gz file_format=myjsonformat on_error='skip_file';

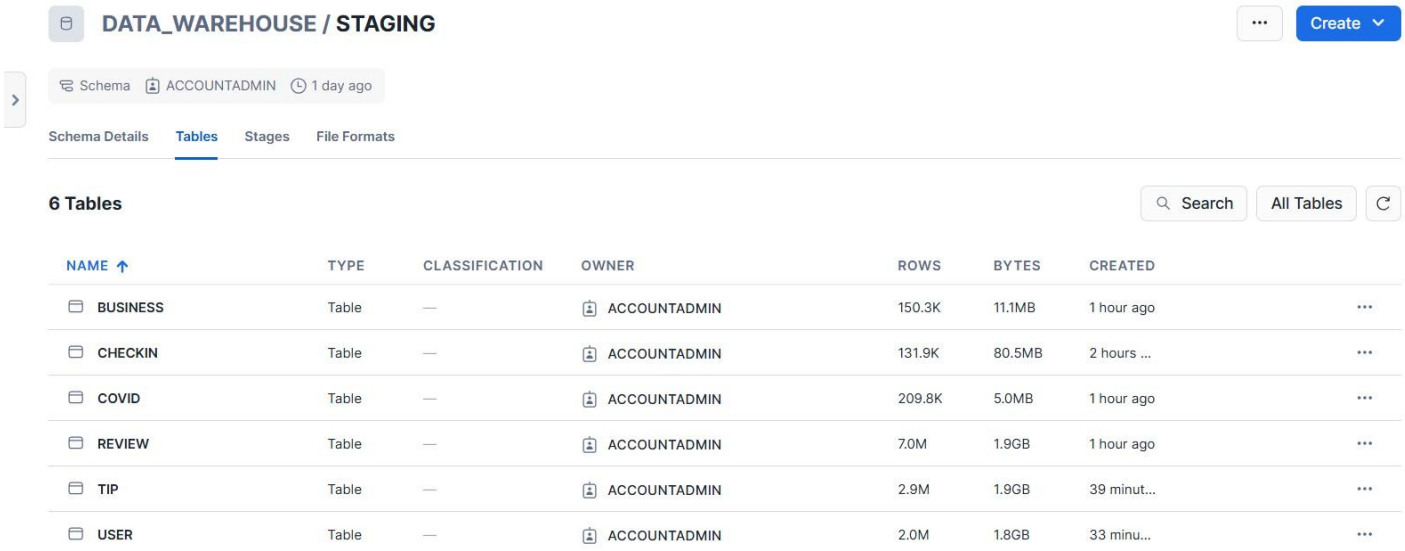
--checkin
create table checkin (usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_checkin.json @my_json_stage auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_checkin.json.gz file_format=myjsonformat on_error='skip_file';

--review
create table review (usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_review.json @my_json_stage auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_review.json.gz file_format=myjsonformat on_error='skip_file';

--covid
create table covid (usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_covid.json @my_json_stage auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_covid.json.gz file_format=myjsonformat on_error='skip_file';

--tip
create table tip(usersjson variant);
--To upload data from your local computer to the Staging Area
put file:///Users/HP/Desktop/warehouseproject/yelp_academic_dataset_tip.json @my_json_stage auto_compress=true;
--Finally copy the data you just uploaded to the staging area into the table
copy into review from @my_json_stage/yelp_academic_dataset_tip.json.gz file_format=myjsonformat on_error='skip_file';

5. Screenshot showing the queries were used successfully to transform staging data from a single JSON structure into multiple columns for ODS



6. Screenshot showing different sizes/row_counts of raw, staging, and ODS tables in database

	A	B	C	D	E
1	**TABLE_NAME**	**RAW**	**STAGING**	**ODS**	
2					
3	BISINESS	113.0 MB	11.0 MB	11.5 MB	
4	COVID	5.0 MB	5.0 MB	5.0 MB	
5	USER	3.13 GB	1.8 GB	1.8 MB	
6	TIP	172 MB	46.1 MB	44.8 MB	
7	CHECKIN	273 MB	80.5 MB	82.5 MB	
8	TEMPEARATURE	797 KB	210.0 KB	173.0 KB	
9	REVIEW	4.97 GB	1.9 GB	1.9 MB	
10	PRECIPITATION	515 KB	146.0 KB	213.5 KB	
11					

7. SQL queries that integrate the climate and Yelp datasets

```
SELECT temp.date_t AS date_temperature,
 temp.min_temp,
 temp.max_temp,
 temp.normal_max_temp,
 temp.normal_min_temp,
 p.date_p AS precipitation_date,
 p.precipitation,
 p.precipitation_normal,
 r.review_id,
```

r.date AS review_date,
r.stars AS review_stars,
r.text AS review_text,
r.cool AS reveiw_cool,
r.useful AS review_useful,
r.funny AS review_funny,
t.compliment_count,
t.text AS tip_text,
t.tip_date AS tip_date,
b.business_id,
b.address,
b.name AS business_name,
b.categories,
b.city,
b.postal_code,
b.review_count AS business_review_count,
b.attributes,
b.is_open,
b.state,
b.hours,
b.latitude,
b.longitude,
b.stars AS business_stars ,
c.highlights,
c.delivery_or_takeout,
c.grubhub_enabled,
c.call_to_action_enabled,
c.request_a_quote_enabled,
c.covid_banner,
c.temporary_closed_until,
c.virtual_services_offered,
u.user_id,
u.name AS user_name,
u.review_count AS user_review_count,
u.yelping_since,
u.friends,
u.useful AS user_useful,
u.funny AS user_funny,
u.cool AS user_cool,
u.fans,
u.elite,
u.average_stars AS user_average_stars,
u.compliment_hot,
u.compliment_more,
u.compliment_profile,
u.compliment_cute,
u.compliment_list,
u.average_stars,
u.compliment_plain,

```
u.compliment_funny,
u.compliment_cool,
u.compliment_writer,
u.compliment_photos

FROM precipitation_d AS p
JOIN review_d AS r
ON r.date = p.date_p
JOIN temperature_d AS temp
ON temp.date_t = r.date
JOIN business_d AS b
ON b.business_id = r.business_id
JOIN covid_d AS c
ON b.business_id = c.business_id
JOIN checkin_d AS ck
ON b.business_id = ck.business_id
JOIN tip_d AS t
ON b.business_id = t.business_id
JOIN user_d AS u
ON u.user_id = r.user_id;

---check
--integrate datasets
SELECT *
FROM precipitation_d AS p
JOIN review_d AS r
ON r.date = p.date_p
JOIN temperature_d AS temp
ON temp.date_t = r.date
JOIN business_d AS b
ON b.business_id = r.business_id
JOIN covid_d AS c
ON b.business_id = c.business_id
JOIN checkin_d AS ck
ON b.business_id = ck.business_id
JOIN tip_d AS t
ON b.business_id = t.business_id
JOIN user_d AS u
ON u.user_id = r.user_id;
```

8. Screenshot showing evidence that the SQL queries managed to integrate the datasets

DATA_WAREHOUSE.ODS Settings Code Versions

```
--integrate datasets
SELECT *
FROM precipitation_d AS p
JOIN review_d AS r
ON r.date = p.date_p
JOIN temperature_d AS temp
ON temp.date_t = r.date
JOIN business_d AS b
ON b.business_id = r.business_id
JOIN covid_d AS c
ON b.business_id = c.business_id
JOIN checkin_d AS ck
ON b.business_id = ck.business_id
JOIN tip_d AS t
ON b.business_id = t.business_id
JOIN user_d AS u
ON u.user_id = r.user_id;
```

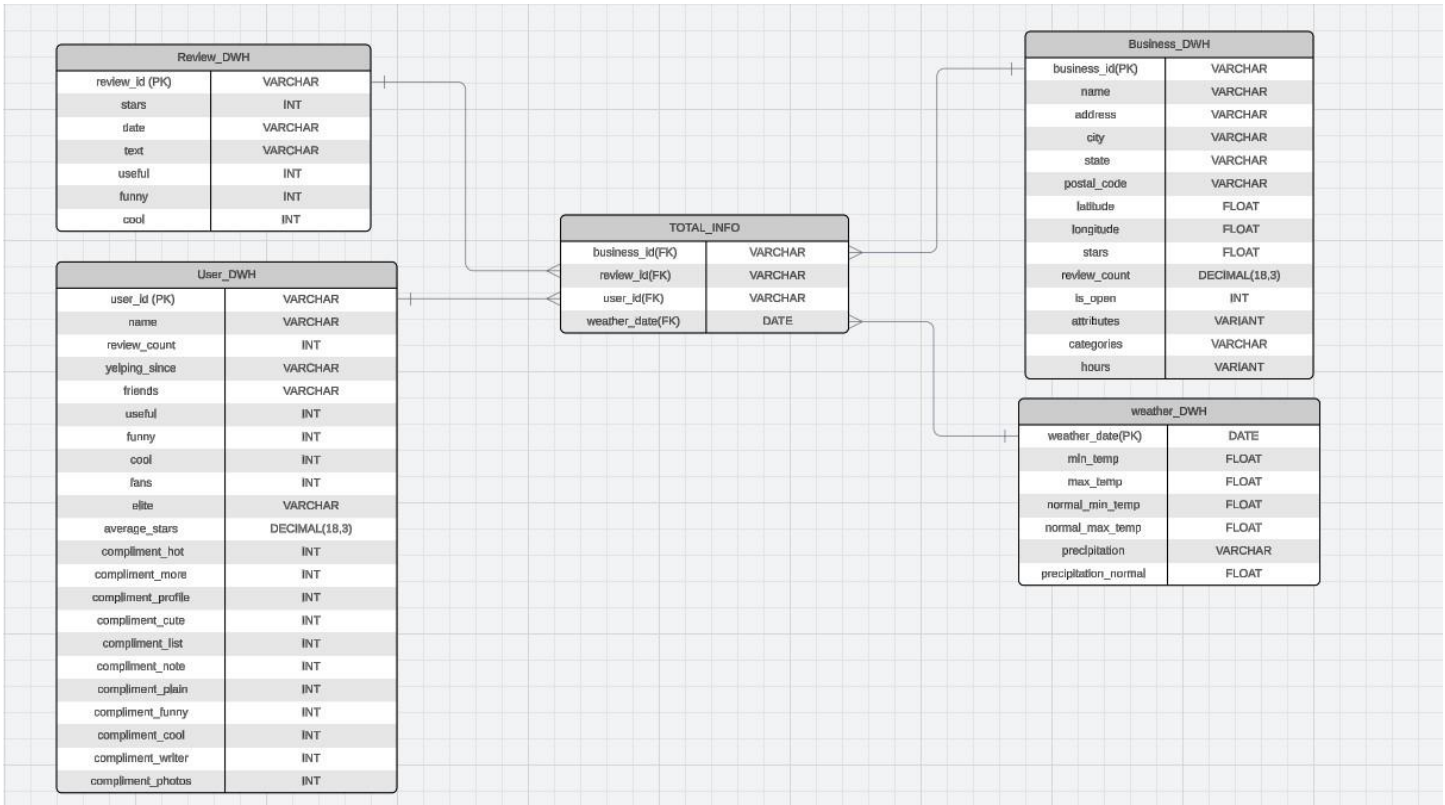
Results Chart

	DATE_P	PRECIPITATION	PRECIPITATION_NORMAL	REVIEW_ID	USER_ID	BUSINESS_ID	STARS	DATE	TEXT
1	2017-09-01	1.92	2.82	cF0KcdYBhJs7wxuwt6qn_Q	AFvZ_7F_ejlCf7AdyxqviA	gPxoK-31HK-IRc4WAX7kxw	5	2017-09-01 15:59:47	Georgia
2	2017-09-01	1.92	2.82	cF0KcdYBhJs7wxuwt6qn_Q	AFvZ_7F_ejlCf7AdyxqviA	gPxoK-31HK-IRc4WAX7kxw	5	2017-09-01 15:59:47	Georgia
3	2017-09-01	1.92	2.82	cF0KcdYBhJs7wxuwt6qn_Q	AFvZ_7F_ejlCf7AdyxqviA	gPxoK-31HK-IRc4WAX7kxw	5	2017-09-01 15:59:47	Georgia
4	2017-09-01	1.92	2.82	cF0KcdYBhJs7wxuwt6qn_Q	AFvZ_7F_ejlCf7AdyxqviA	gPxoK-31HK-IRc4WAX7kxw	5	2017-09-01 15:59:47	Georgia
5	2017-09-01	1.92	2.82	cF0KcdYBhJs7wxuwt6qn_Q	AFvZ_7F_ejlCf7AdyxqviA	gPxoK-31HK-IRc4WAX7kxw	5	2017-09-01 15:59:47	Georgia

Query Details
Query duration 8.7s
Rows 3.8K
Query ID 01ba1acf-3201-6807-00...
Show more

C. Data Warehouse (DWH)

1. Diagram of star schema with several dimensions and a fact table that connects dimensions



2. SQL queries that moves the data from ODS to DWH

```
--business_DWH Table
CREATE TABLE BUSINESS_DWH(
 business_id VARCHAR PRIMARY KEY,
 name VARCHAR,
 address VARCHAR,
 city VARCHAR,
 state VARCHAR,
 postal_code VARCHAR,
 latitude FLOAT,
 longitude FLOAT,
 stars FLOAT,
 review_count DECIMAL(18,3),
 is_open INT,
 attributes VARIANT,
 categories VARCHAR,
 hours VARIANT
);
INSERT INTO BUSINESS_DWH
SELECT
    business_id,
    name,
    address,
    city,
    state,
    postal_code,
    latitude,
    longitude,
    stars,
    review_count,
    is_open,
    attributes,
    categories,
    hours
FROM "DATA_WAREHOUSE"."ODS".business_d;

--user_DWH Table
CREATE TABLE USER_DWH(
 user_id VARCHAR PRIMARY KEY,
 name VARCHAR,
 review_count INT,
 yelping_since VARCHAR,
 friends VARCHAR,
 useful INT,
 funny INT,
 cool INT,
 fans INT,
 elite VARCHAR,
 average_stars DECIMAL(18,3),
```

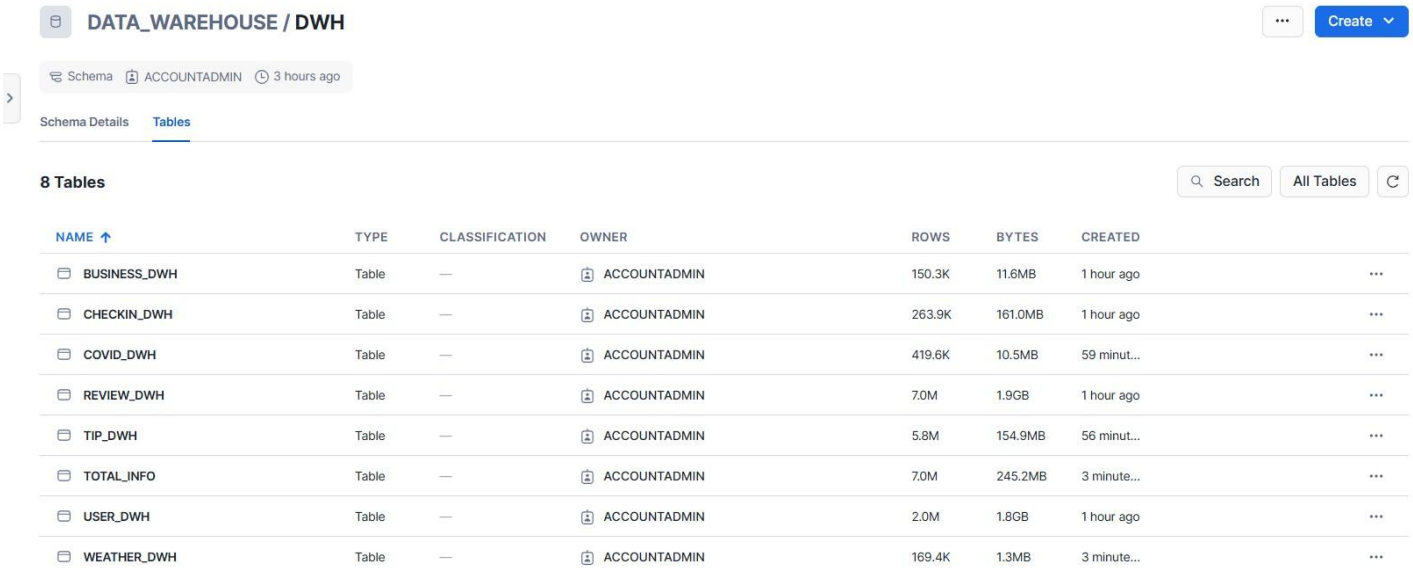
```
 compliment_hot INT,
 compliment_more INT,
 compliment_profile INT,
 compliment_cute INT,
 compliment_list INT,
 compliment_note INT,
 compliment_plain INT,
 compliment_funny INT,
 compliment_cool INT,
 compliment_writer INT,
 compliment_photos INT
);
INSERT INTO USER_DWH
SELECT
    user_id,
    name,
    review_count,
    yelping_since,
    friends,
    useful,
    funny,
    cool,
    fans,
    elite,
    average_stars,
    compliment_hot,
    compliment_more,
    compliment_profile,
    compliment_cute,
    compliment_list,
    compliment_note,
    compliment_plain,
    compliment_funny,
    compliment_cool,
    compliment_writer,
    compliment_photos
FROM "DATA_WAREHOUSE"."ODS".user_d;

--checkin_DWH Table
CREATE TABLE checkin_DWH (
    checkin_date VARCHAR,
    business_id VARCHAR
);
INSERT INTO checkin_DWH
SELECT
    checkin_date,
    business_id
FROM "DATA_WAREHOUSE"."ODS".checkin_D;
```

```
--review_DWH Table
CREATE TABLE review_DWH (
    review_id VARCHAR PRIMARY KEY,
    stars INT,
    date VARCHAR,
    text VARCHAR,
    useful INT,
    funny INT,
    cool INT
);
INSERT INTO review_DWH
SELECT
    review_id,
    stars,
    date,
    text,
    useful,
    funny,
    cool
FROM "DATA_WAREHOUSE"."ODS".review_D;

--covid_DWH Table
CREATE TABLE covid_DWH (
    business_id VARCHAR,
    highlights VARCHAR,
    delivery_or_takeout VARIANT,
    grubhub_enabled VARIANT,
    call_to_action_enabled VARIANT,
    request_a_quote_enabled VARIANT,
    covid_banner VARIANT,
    temporary_closed_until VARIANT,
    virtual_services_offered VARIANT
);
INSERT INTO covid_DWH
SELECT
    business_id,
    highlights,
    delivery_or_takeout,
    grubhub_enabled,
    call_to_action_enabled,
    Request_a_quote_enabled,
    covid_banner,
    temporary_closed_until,
    virtual_services_offered
FROM "DATA_WAREHOUSE"."ODS".covid_d;

--tip_DWH Table
CREATE TABLE tip_DWH (
    text VARCHAR,
```

```
    tip_date VARCHAR,
    compliment_count INT,
    business_id VARCHAR,
    user_id VARCHAR
);
INSERT INTO tip_DWH
SELECT
    text,
    tip_date,
    compliment_count,
    business_id,
    user_id
FROM "DATA_WAREHOUSE"."ODS".tip_d;

--weather_DWH
CREATE TABLE weather_DWH(
   weather_date DATE PRIMARY KEY,
   min_temp FLOAT,
   max_temp FLOAT,
   normal_min_temp FLOAT,
   normal_max_temp FLOAT,
   precipitation VARCHAR,
   precipitation_normal FLOAT
);

INSERT INTO weather_DWH
SELECT
        t.date_t,
        t.min_temp,
        t.max_temp,
        t.normal_min_temp,
        t.normal_max_temp,
        p.precipitation,
        p.precipitation_normal
FROM "DATA_WAREHOUSE"."ODS".temperature_d AS t
JOIN "DATA_WAREHOUSE"."ODS".precipitation_d AS p
ON t.date_t = p.date_p;

--fact table
create table TOTAL_INFO(
   business_id VARCHAR,
   review_id VARCHAR,
   user_id VARCHAR,
   weather_date DATE,
   FOREIGN KEY (business_id) REFERENCES business_DWH(business_id),
   FOREIGN KEY (review_id) REFERENCES review_DWH(review_id),
   FOREIGN KEY (user_id) REFERENCES user_DWH(user_id),
   FOREIGN KEY (weather_date) REFERENCES weather_DWH(weather_date)
);
```

```
INSERT INTO TOTAL_INFO
SELECT
    r.review_id,
    u.user_id,
    b.business_id,
    r.date
FROM "DATA_WAREHOUSE"."ODS".review_d AS r
JOIN "DATA_WAREHOUSE"."ODS".business_d AS b
ON b.business_id  = r.business_id
JOIN "DATA_WAREHOUSE"."ODS".user_d AS u
ON r.user_id = u.user_id;
```

3. Screenshot showing evidence that the SQL queries managed to move the data from ODS to DWH

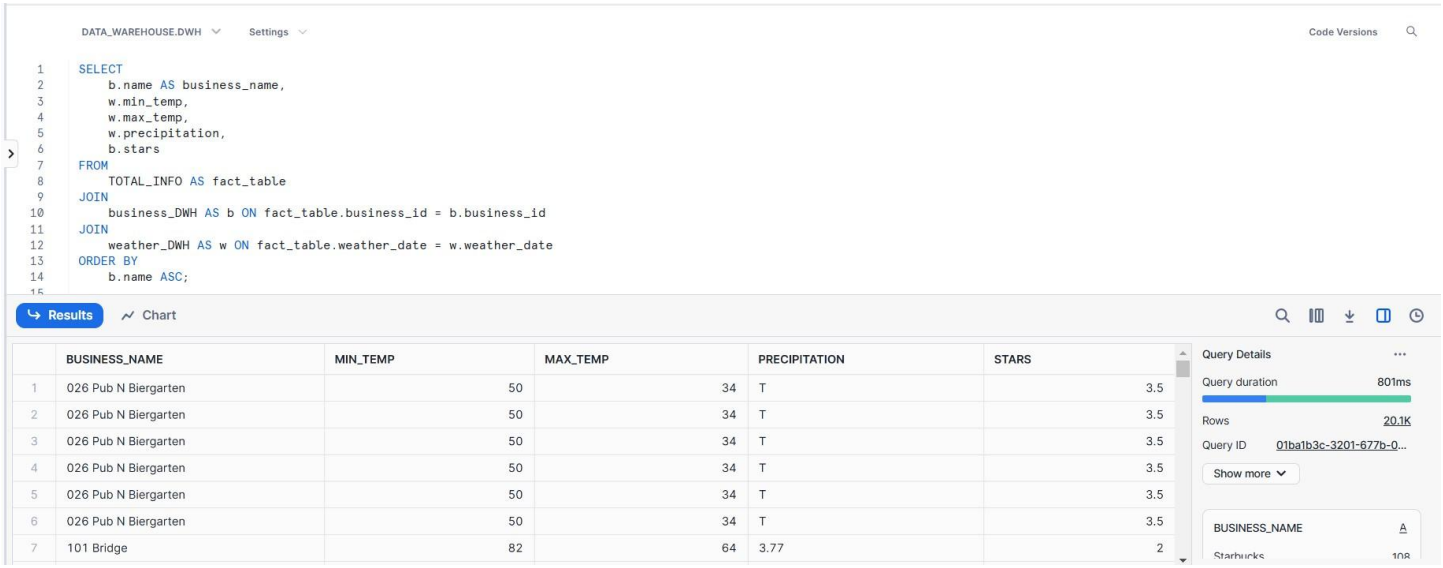


4. SQL queries that produce a report showing the business name, temperature, precipitation, and ratings

```
SELECT
    b.name AS business_name,
    w.min_temp,
    w.max_temp,
    w.precipitation,
    b.stars
FROM
    TOTAL_INFO AS fact_table
JOIN
    business_DWH AS b ON fact_table.business_id = b.business_id
```

```
JOIN
    weather_DWH AS w ON fact_table.weather_date = w.weather_date
ORDER BY
    b.name ASC;
```

5. Screenshot showing the report produced by the SQL queries above



DATA_WAREHOUSE.DWH Settings Code Versions

```
SELECT
    b.name AS business_name,
    w.min_temp,
    w.max_temp,
    w.precipitation,
    b.stars
FROM
    TOTAL_INFO AS fact_table
JOIN
    business_DWH AS b ON fact_table.business_id = b.business_id
JOIN
    weather_DWH AS w ON fact_table.weather_date = w.weather_date
ORDER BY
    b.name ASC;
```

Results Chart

	BUSINESS_NAME	MIN_TEMP	MAX_TEMP	PRECIPITATION	STARS
1	026 Pub N Biergarten	50	34	T	3.5
2	026 Pub N Biergarten	50	34	T	3.5
3	026 Pub N Biergarten	50	34	T	3.5
4	026 Pub N Biergarten	50	34	T	3.5
5	026 Pub N Biergarten	50	34	T	3.5
6	026 Pub N Biergarten	50	34	T	3.5
7	101 Bridge	82	64	3.77	2

Query Details
Query duration 801ms
Rows 20.1K
Query ID 01ba1b3c-3201-677b-0...
Show more
BUSINESS_NAME A
Starbucks 108