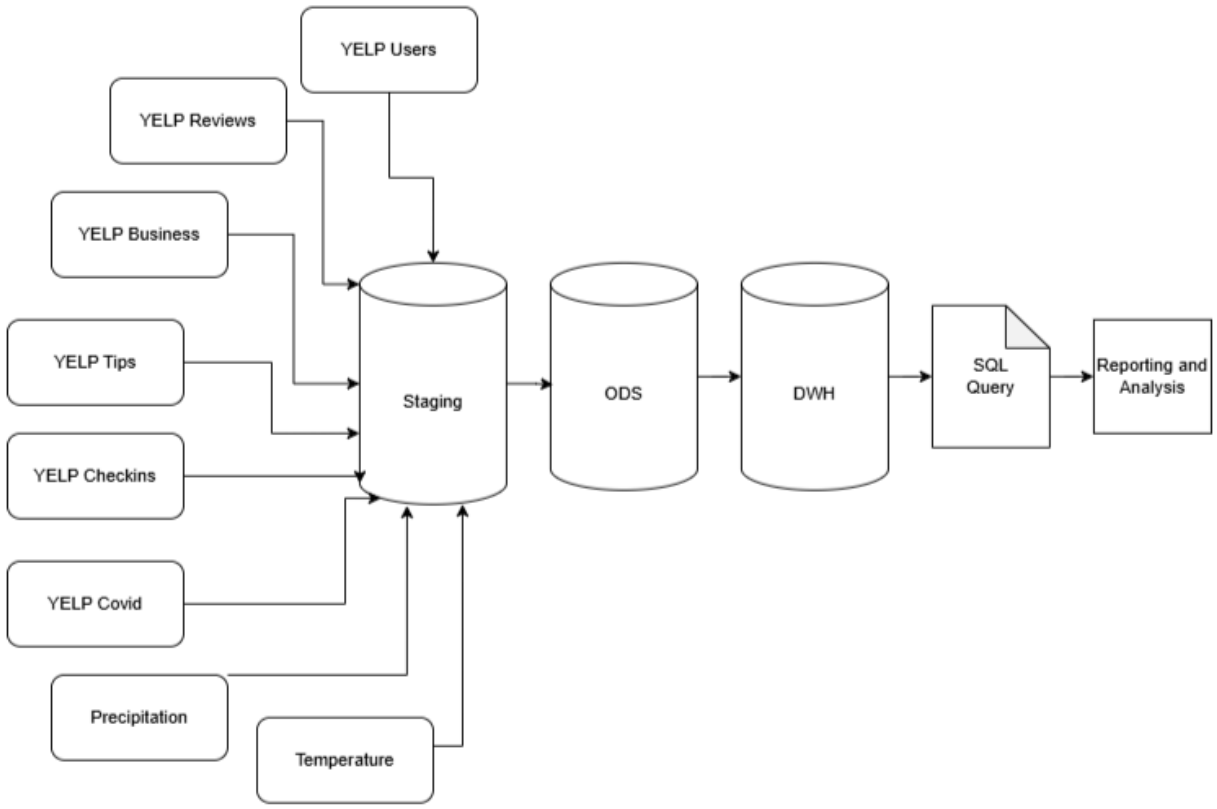
Submission Document - Design a Data Warehouse for Reporting and OLAP

Please include all of the required screenshots and SQL queries in this document. Resize the images as necessary to ensure the texts are readable.

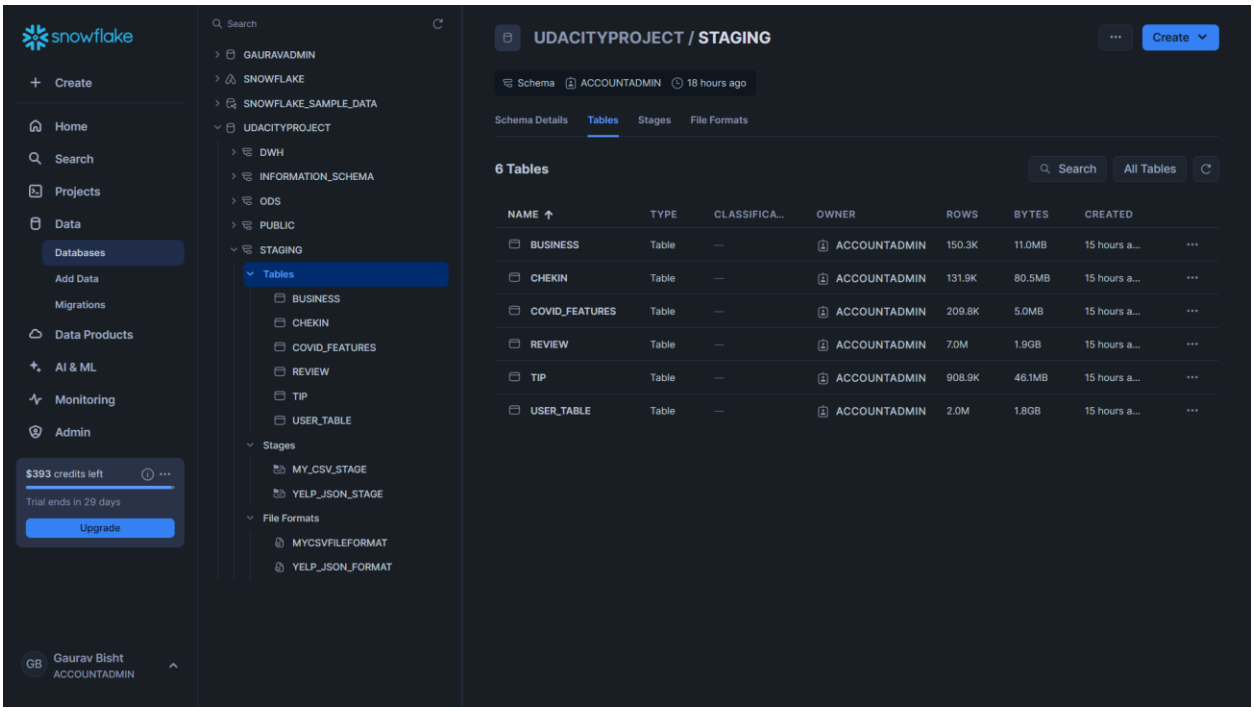
A. Staging

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



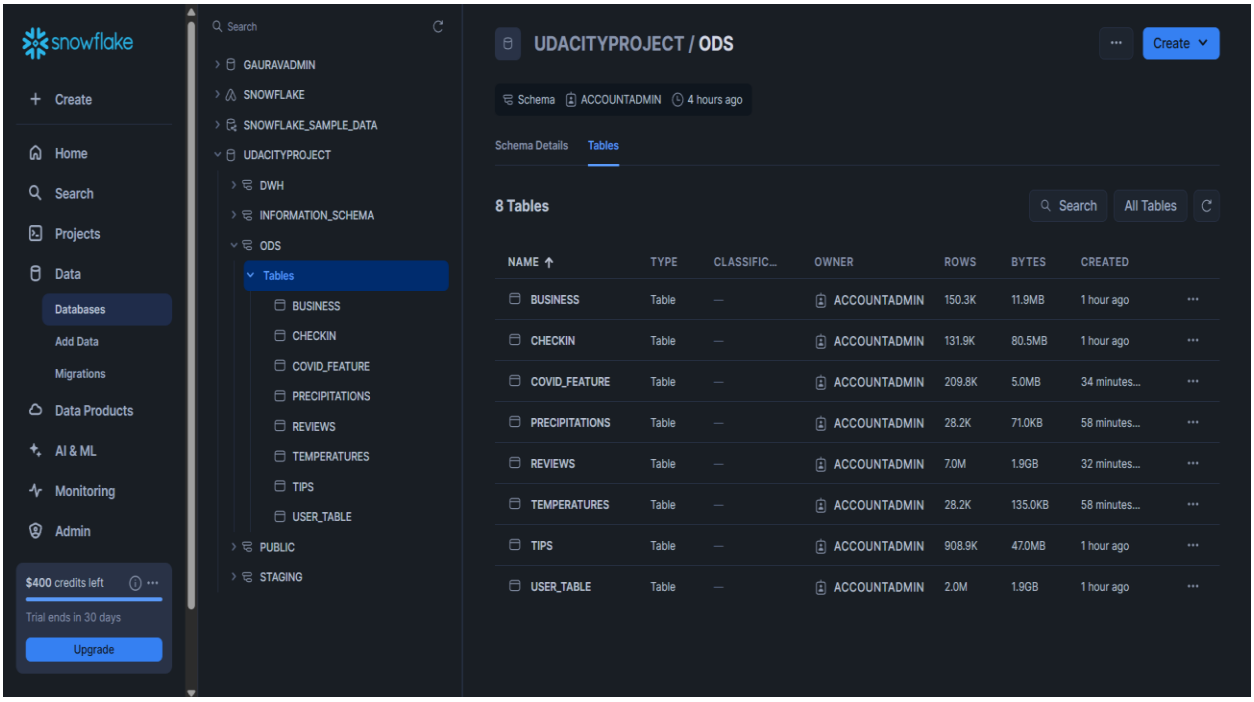
2. Screenshot showing the tables in the staging schema after extracting 6 Yelp files

The screenshot should have 6 tables with the correct respective row counts.



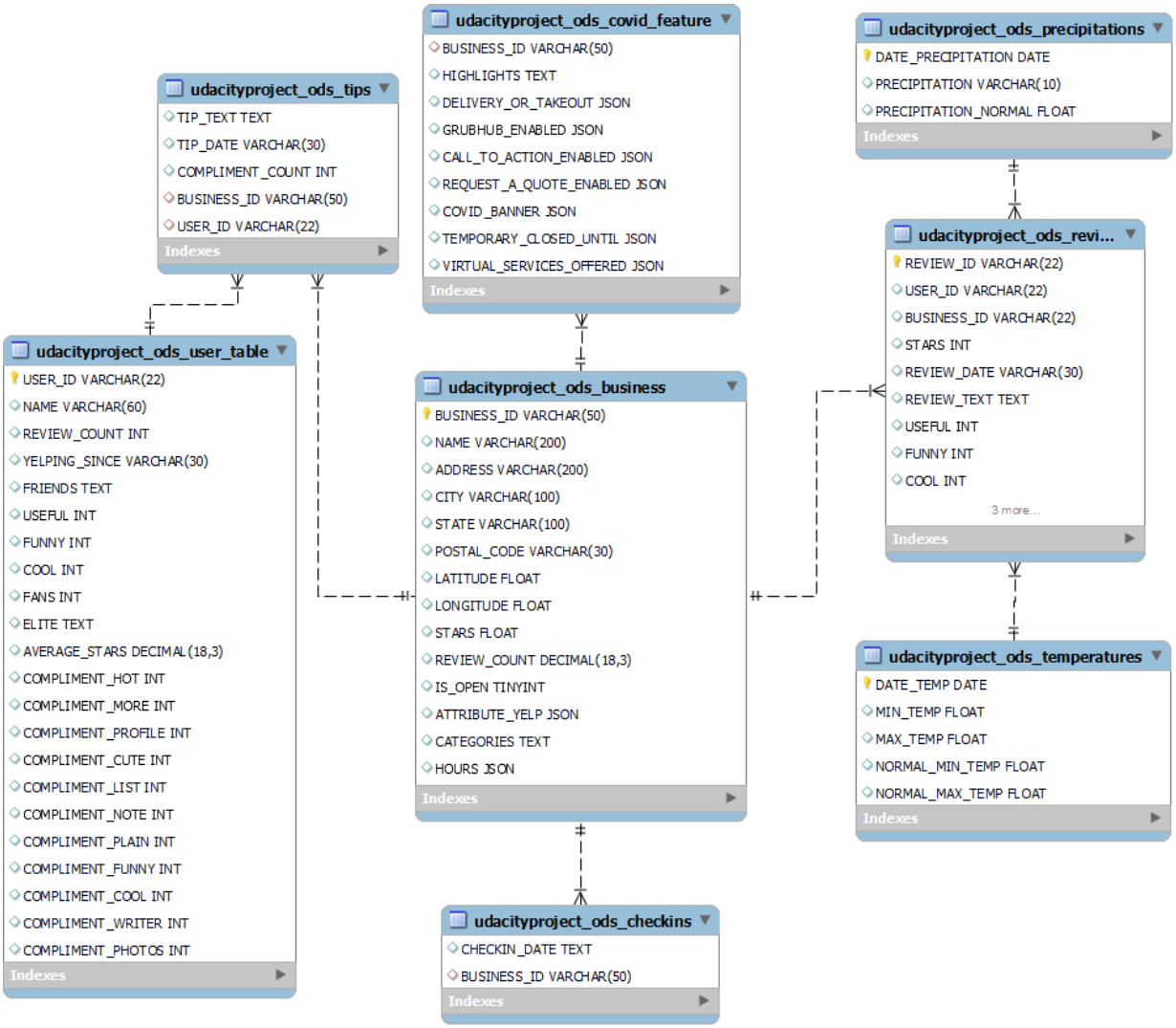
3. Screenshot showing the tables after extracting 2 files into the staging schema

The screenshot should have two tables - temperature and precipitation.



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

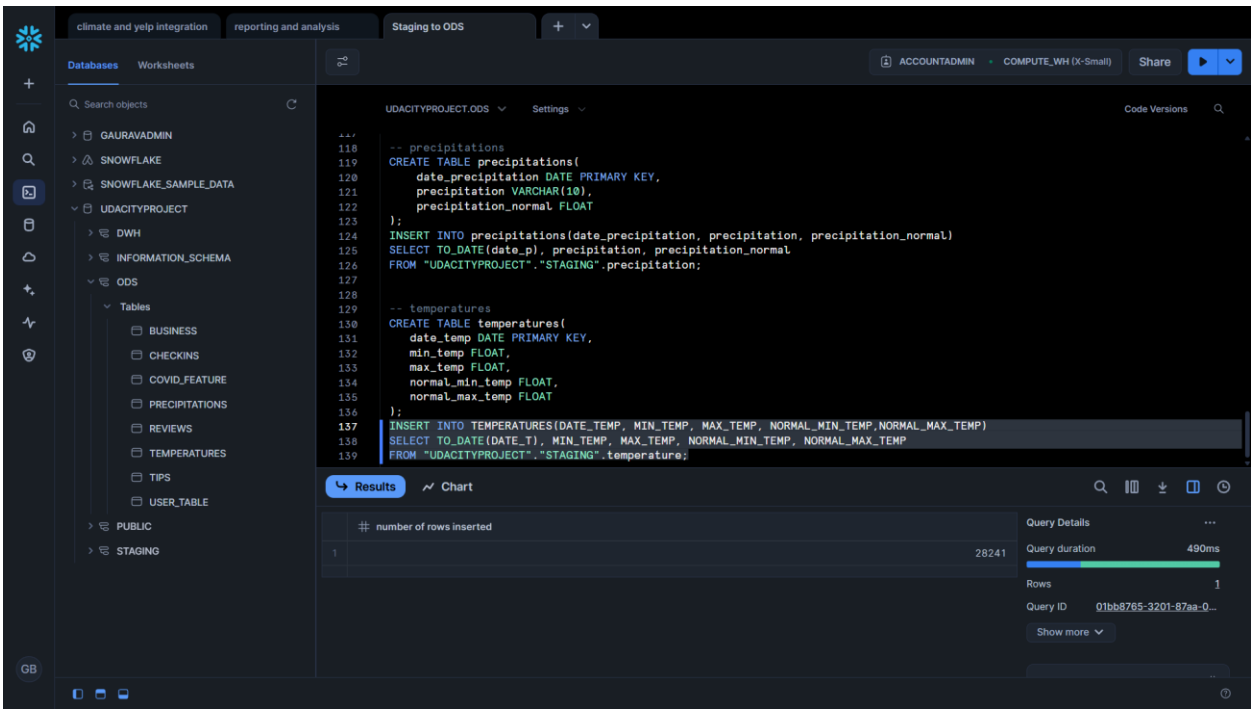


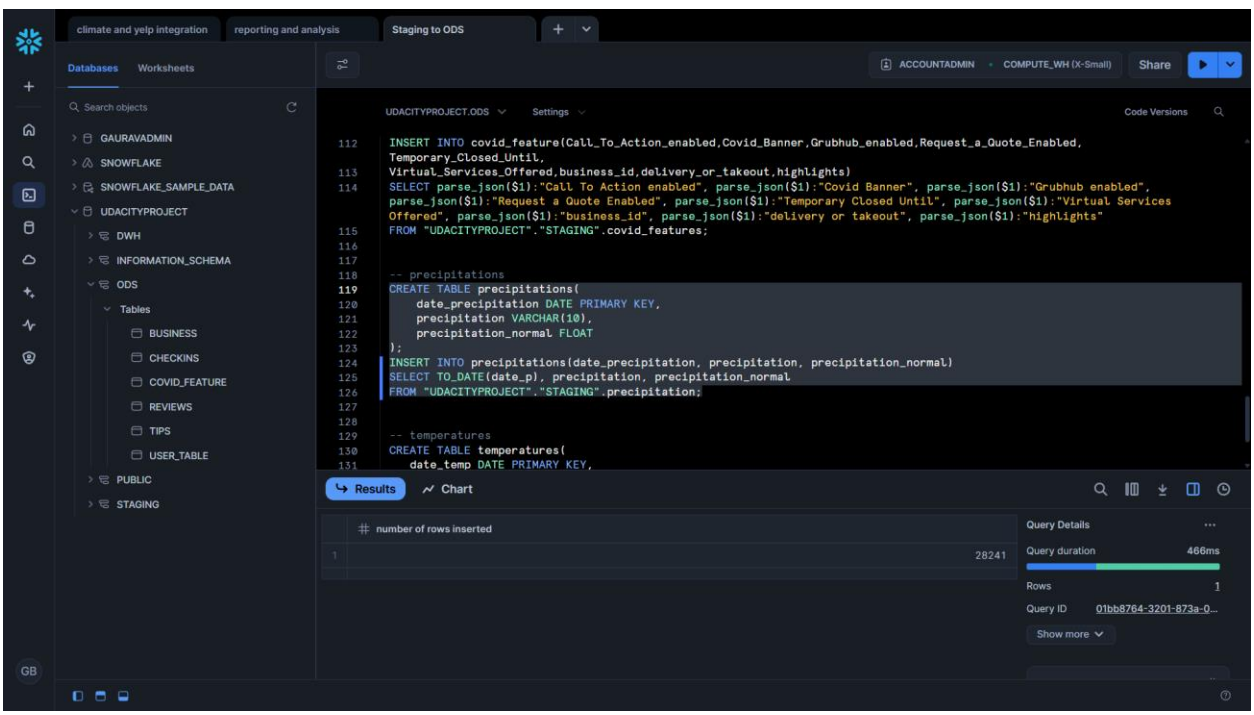
2. SQL queries that transform staging to ODS

```
-- precipitations
CREATE TABLE precipitations(
    date_precipitation DATE PRIMARY KEY,
    precipitation VARCHAR(10),
    precipitation_normal FLOAT
```

```sql
);
INSERT INTO precipitations(date_precipitation, precipitation,
precipitation_normal)
SELECT TO_DATE(date_p), precipitation, precipitation_normal
FROM "UDACITYPROJECT"."STAGING".precipitation;


-- temperatures
CREATE TABLE temperatures(
   date_temp DATE PRIMARY KEY,
   min_temp FLOAT,
   max_temp FLOAT,
   normal_min_temp FLOAT,
   normal_max_temp FLOAT
);
INSERT INTO TEMPERATURES(DATE_TEMP, MIN_TEMP, MAX_TEMP,
NORMAL_MIN_TEMP,NORMAL_MAX_TEMP)
SELECT TO_DATE(DATE_T), MIN_TEMP, MAX_TEMP, NORMAL_MIN_TEMP,
NORMAL_MAX_TEMP
FROM "UDACITYPROJECT"."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

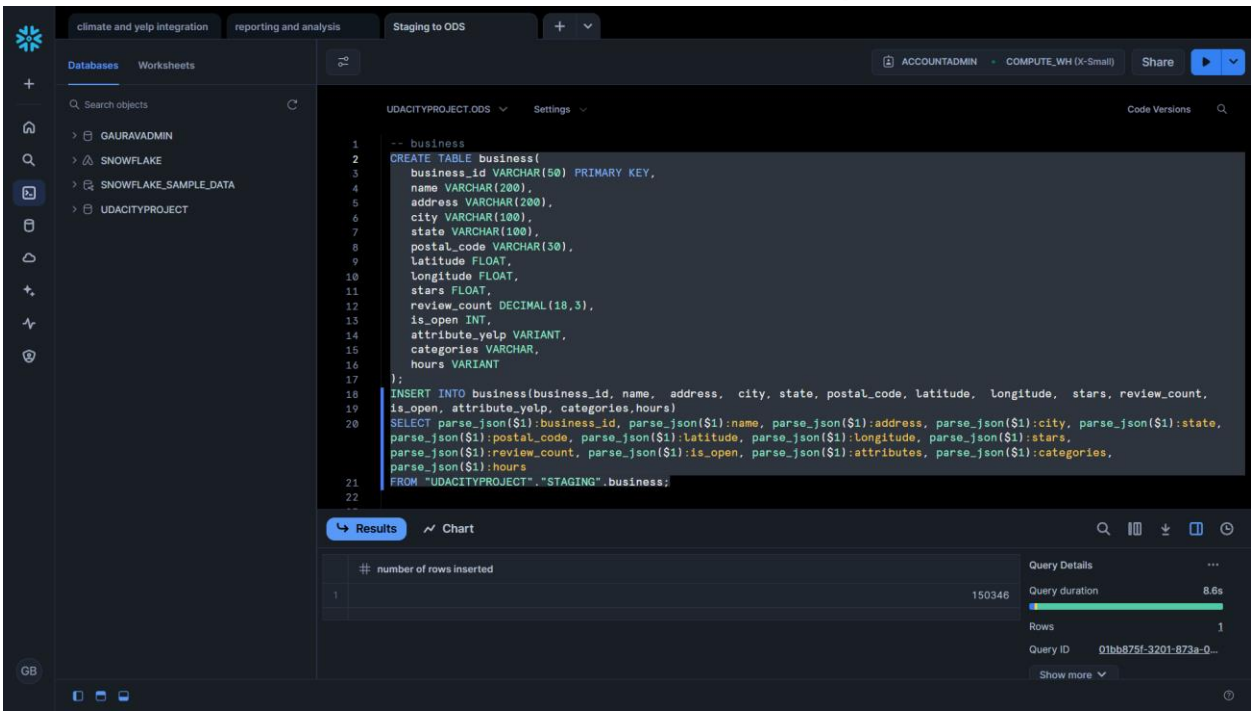
```
-- business
CREATE TABLE business(
   business_id VARCHAR(50) PRIMARY KEY,
   name VARCHAR(200),
   address VARCHAR(200),
   city VARCHAR(100),
   state VARCHAR(100),
   postal_code VARCHAR(30),
   latitude FLOAT,
   longitude FLOAT,
   stars FLOAT,
   review_count DECIMAL(18,3),
   is_open INT,
   attribute_yelp VARIANT,
   categories VARCHAR,
   hours VARIANT
);
INSERT INTO business(business_id, name,  address,  city, state,
postal_code, latitude,  longitude,  stars, review_count,
is_open, attribute_yelp, categories,hours)
SELECT parse_json($1):business_id, parse_json($1):name,
parse_json($1):address, parse_json($1):city, parse_json($1):state,
parse_json($1):postal_code, parse_json($1):latitude,
parse_json($1):longitude, parse_json($1):stars,
parse_json($1):review_count, parse_json($1):is_open,
parse_json($1):attributes, parse_json($1):categories, parse_json($1):hours
FROM "UDACITYPROJECT"."STAGING".business;


-- user_table
CREATE TABLE user_table(
    user_id VARCHAR(22) PRIMARY KEY,
    name VARCHAR(60),
    review_count INT,
    yelping_since VARCHAR(30),
    friends VARCHAR,
    useful INT,
    funny INT,
```

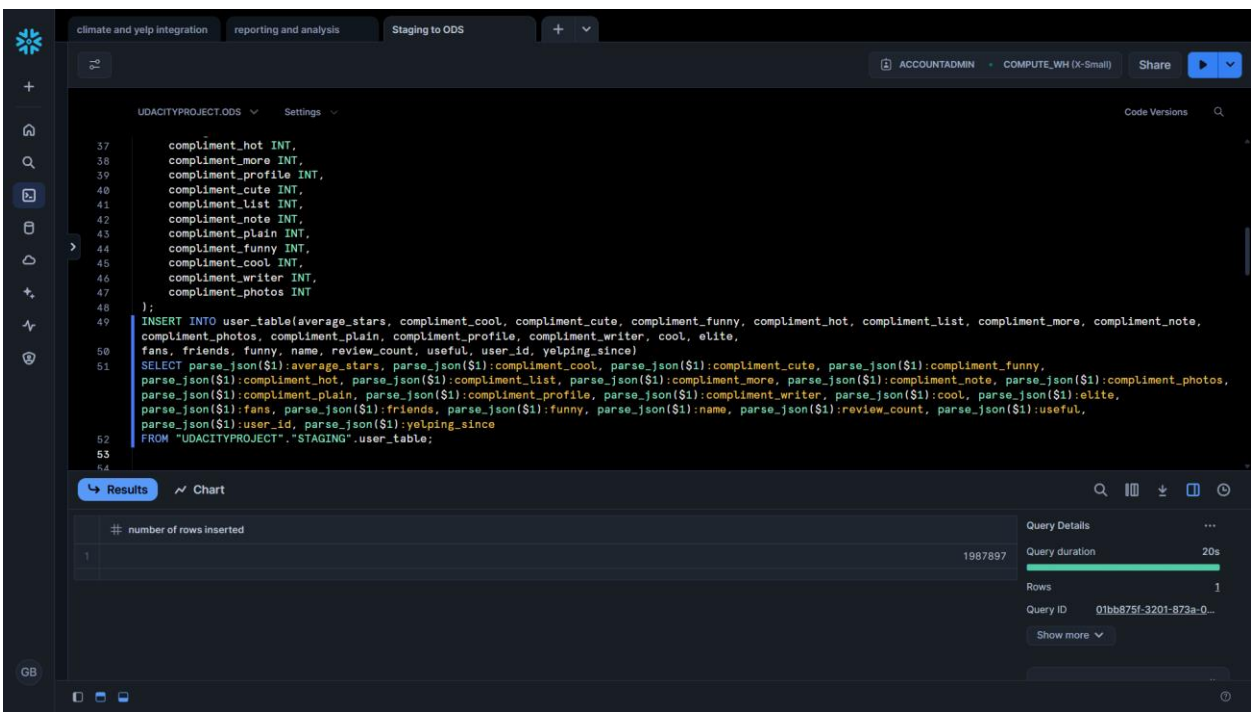
```sql
    cool INT,
    fans INT,
    elite VARCHAR,
    average_stars DECIMAL(18,3),
    compliment_hot INT,
    compliment_more INT,
    compliment_profile INT,
    compliment_cute INT,
    compliment_list INT,
    compliment_note INT,
    compliment_plain INT,
    compliment_funny INT,
    compliment_cool INT,
    compliment_writer INT,
    compliment_photos INT
);
INSERT INTO user_table(average_stars, compliment_cool, compliment_cute,
compliment_funny, compliment_hot, compliment_list, compliment_more,
compliment_note,  compliment_photos, compliment_plain, compliment_profile,
compliment_writer, cool, elite,
fans, friends, funny, name, review_count, useful, user_id, yelping_since)
SELECT parse_json($1):average_stars, parse_json($1):compliment_cool,
parse_json($1):compliment_cute, parse_json($1):compliment_funny,
parse_json($1):compliment_hot, parse_json($1):compliment_list,
parse_json($1):compliment_more, parse_json($1):compliment_note,
parse_json($1):compliment_photos, parse_json($1):compliment_plain,
parse_json($1):compliment_profile, parse_json($1):compliment_writer,
parse_json($1):cool, parse_json($1):elite, parse_json($1):fans,
parse_json($1):friends, parse_json($1):funny, parse_json($1):name,
parse_json($1):review_count, parse_json($1):useful, parse_json($1):user_id,
parse_json($1):yelping_since
FROM "UDACITYPROJECT"."STAGING".user_table;


-- reviews
CREATE TABLE reviews(
    review_id VARCHAR(22) PRIMARY KEY,
    user_id VARCHAR(22),
    business_id VARCHAR(22),
    stars INT,
    review_date VARCHAR(30),
    review_text VARCHAR,
    useful INT,
```

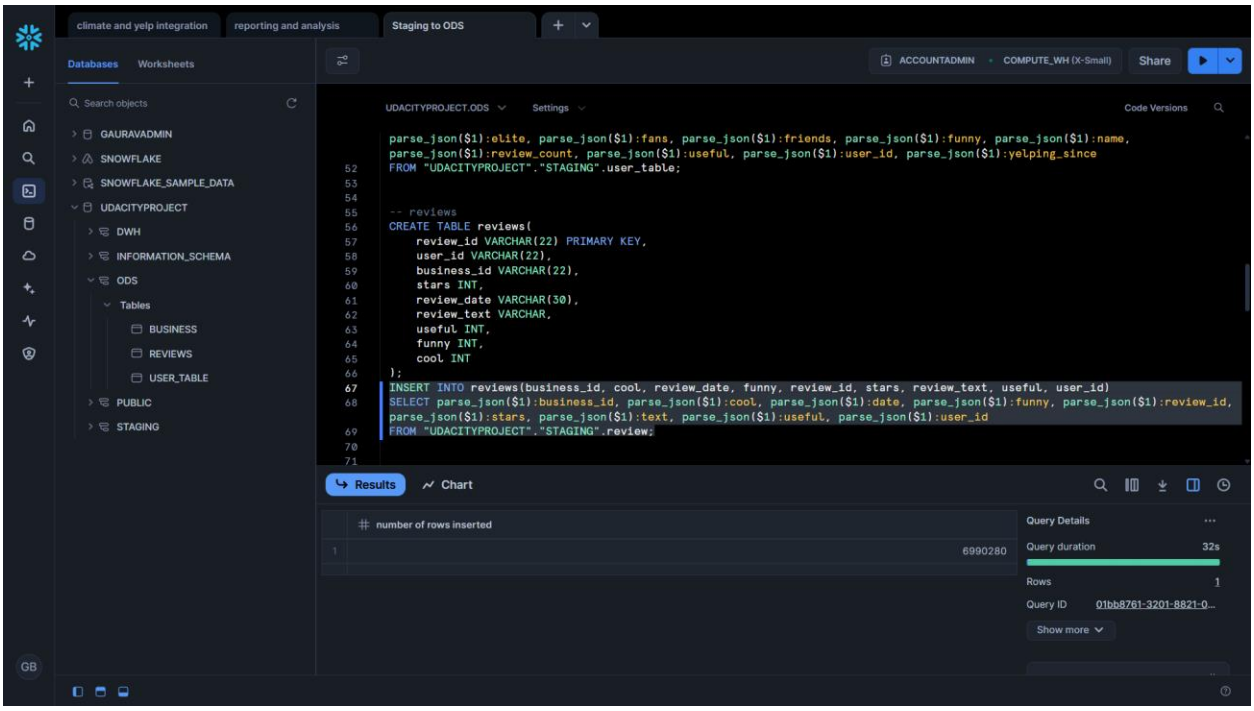
```sql
    funny INT,
    cool INT
);
INSERT INTO reviews(business_id, cool, review_date, funny, review_id,
stars, review_text, useful, user_id)
SELECT parse_json($1):business_id, parse_json($1):cool,
parse_json($1):date, parse_json($1):funny, parse_json($1):review_id,
parse_json($1):stars, parse_json($1):text, parse_json($1):useful,
parse_json($1):user_id
FROM "UDACITYPROJECT"."STAGING".review;


-- checkins
CREATE TABLE checkins(
    checkin_date VARCHAR,
    business_id VARCHAR(50),
    FOREIGN KEY(business_id) REFERENCES business(business_id)
);

INSERT INTO checkins(business_id, checkin_date)
SELECT parse_json($1):business_id, parse_json($1):date
FROM "UDACITYPROJECT"."STAGING".chekin;


-- tips
CREATE TABLE tips(
   tip_text VARCHAR,
   tip_date VARCHAR(30),
   compliment_count INT,
   business_id VARCHAR(50),
   user_id VARCHAR(22),
   FOREIGN KEY(business_id) REFERENCES business(business_id),
   FOREIGN KEY(user_id) REFERENCES user_table(user_id)
);
INSERT INTO tips(business_id, compliment_count, tip_date, tip_text,
user_id)
SELECT parse_json($1):business_id, parse_json($1):compliment_count,
parse_json($1):date, parse_json($1):text, parse_json($1):user_id
FROM "UDACITYPROJECT"."STAGING".tip;


-- covid_feature
CREATE TABLE covid_feature(
```

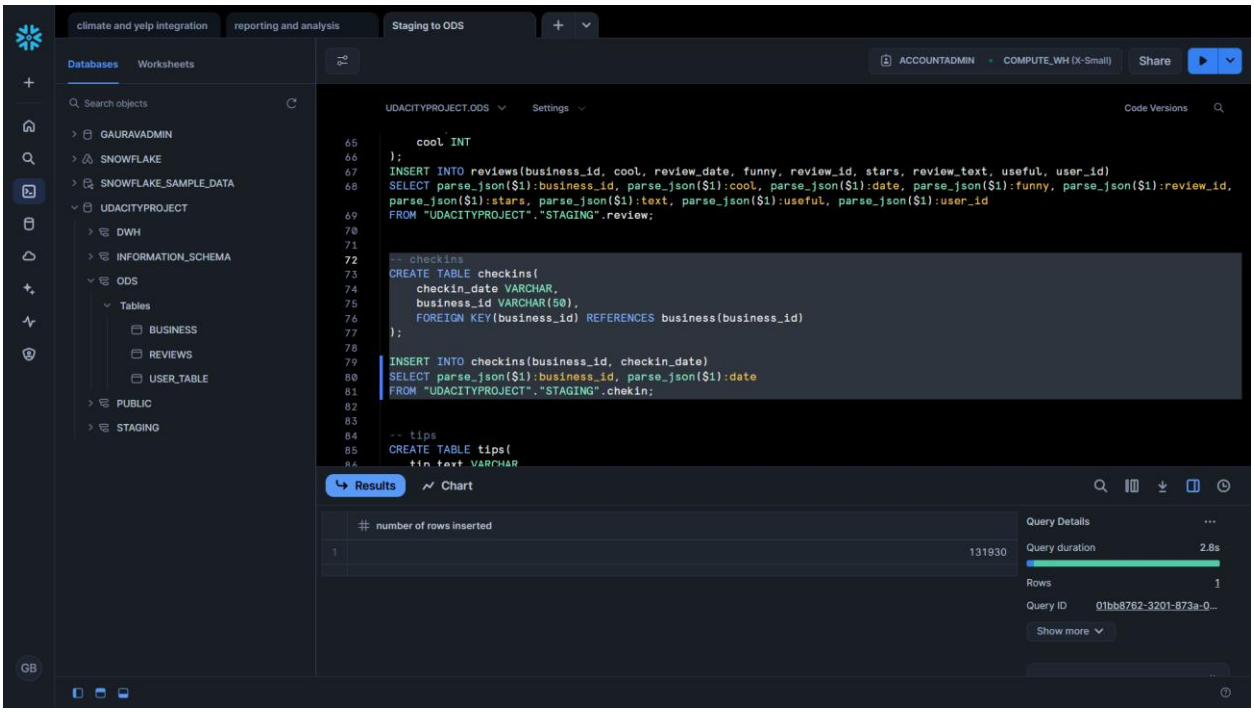
```
    business_id VARCHAR(50),
    highlights VARCHAR,
    delivery_or_takeout VARIANT,
    grubhub_enabled VARIANT,
    call_to_action_enabled VARIANT,
    request_a_quote_enabled VARIANT,
    covid_banner VARIANT,
    temporary_closed_until VARIANT,
    virtual_services_offered VARIANT,
    FOREIGN KEY (business_id) REFERENCES business(business_id)
);
INSERT INTO
covid_feature(Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a
_Quote_Enabled, Temporary_Closed_Until,
Virtual_Services_Offered,business_id,delivery_or_takeout,highlights)
SELECT parse_json($1):"Call To Action enabled", parse_json($1):"Covid
Banner", parse_json($1):"Grubhub enabled", parse_json($1):"Request a Quote
Enabled", parse_json($1):"Temporary Closed Until", parse_json($1):"Virtual
Services Offered", parse_json($1):"business_id", parse_json($1):"delivery
or takeout", parse_json($1):"highlights"
FROM "UDACITYPROJECT"."STAGING".covid_features;
```

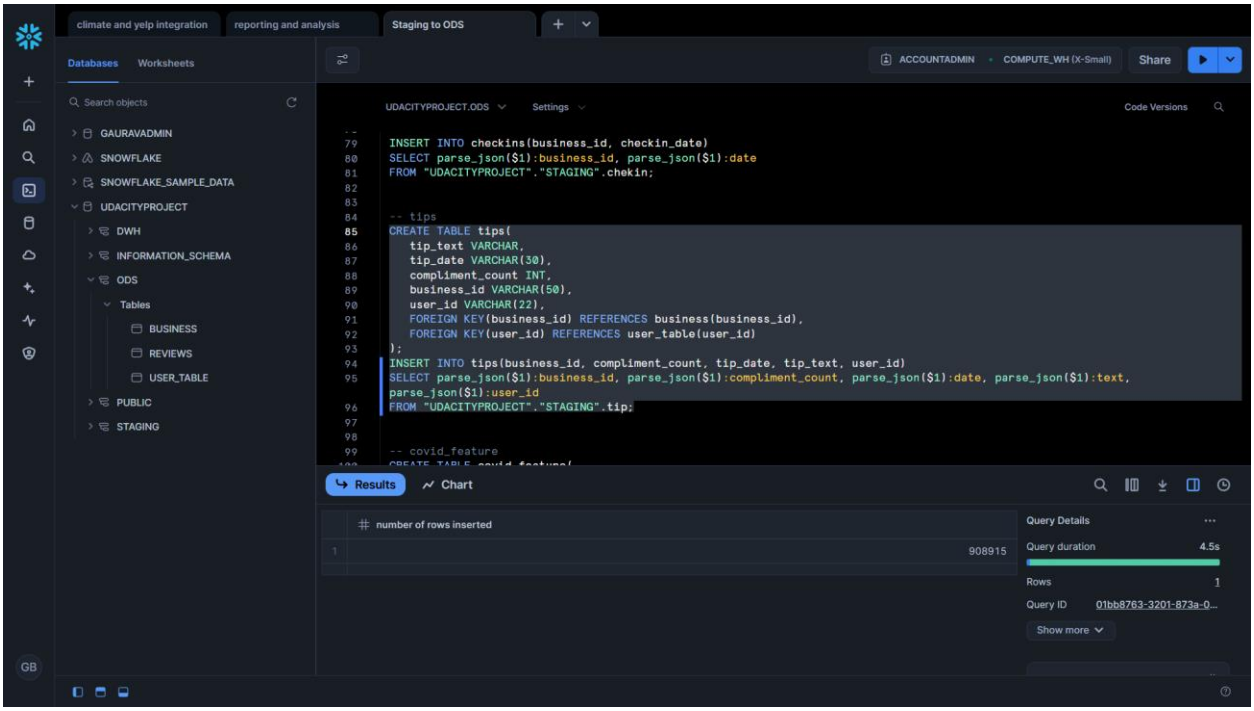
5. Screenshot showing the queries were used successfully to transform staging data from a single JSON structure into multiple columns for ODS

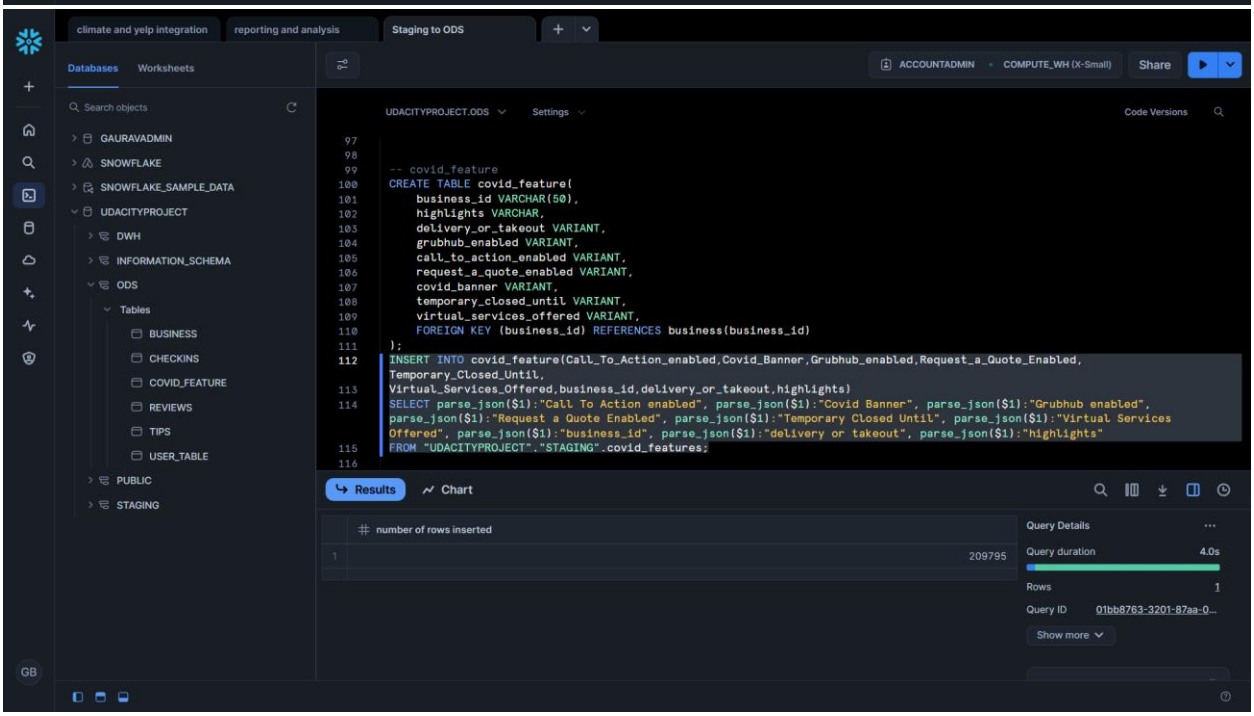


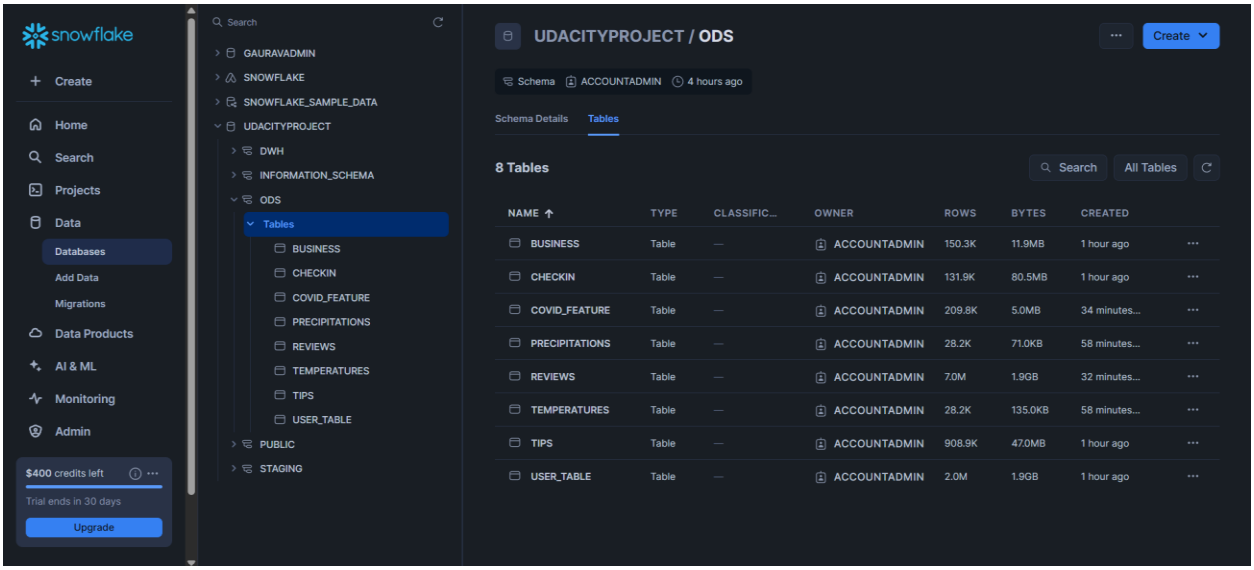



climate and yelp integration
reporting and analysis
Staging to ODS
Databases
Worksheets
Search objects
GAURAVADMIN
SNOWFLAKE
SNOWFLAKE_SAMPLE_DATA
UDACITYPROJECT
DWH
INFORMATION_SCHEMA
ODS
Tables
BUSINESS
REVIEWS
USER_TABLE
PUBLIC
STAGING
ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
UDACITYPROJECT.ODS
Settings
Code Versions
parse_json($1):elite, parse_json($1):fans, parse_json($1):friends, parse_json($1):funny, parse_json($1):name,
parse_json($1):review_count, parse_json($1):useful, parse_json($1):user_id, parse_json($1):yelping_since
FROM "UDACITYPROJECT"."STAGING".user_table;
-- reviews
CREATE TABLE reviews(
review_id VARCHAR(22) PRIMARY KEY,
user_id VARCHAR(22),
business_id VARCHAR(22),
stars INT,
review_date VARCHAR(30),
review_text VARCHAR,
useful INT,
funny INT,
cool INT
);
INSERT INTO reviews(business_id, cool, review_date, funny, review_id, stars, review_text, useful, user_id)
SELECT parse_json($1):business_id, parse_json($1):cool, parse_json($1):date, parse_json($1):funny, parse_json($1):review_id,
parse_json($1):stars, parse_json($1):text, parse_json($1):useful, parse_json($1):user_id
FROM "UDACITYPROJECT"."STAGING".review;
Results
Chart
number of rows inserted
6990280
Query Details
Query duration
32s
Rows
1
Query ID
01bb8761-3201-8821-0...
Show more


climate and yelp integration
reporting and analysis
Staging to ODS
Databases
Worksheets
Search objects
GAURAVADMIN
SNOWFLAKE
SNOWFLAKE_SAMPLE_DATA
UDACITYPROJECT
DWH
INFORMATION_SCHEMA
ODS
Tables
BUSINESS
REVIEWS
USER_TABLE
PUBLIC
STAGING
ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
UDACITYPROJECT.ODS
Settings
Code Versions
cool INT
);
INSERT INTO reviews(business_id, cool, review_date, funny, review_id, stars, review_text, useful, user_id)
SELECT parse_json($1):business_id, parse_json($1):cool, parse_json($1):date, parse_json($1):funny, parse_json($1):review_id,
parse_json($1):stars, parse_json($1):text, parse_json($1):useful, parse_json($1):user_id
FROM "UDACITYPROJECT"."STAGING".review;
-- checkins
CREATE TABLE checkins(
checkin_date VARCHAR,
business_id VARCHAR(50),
FOREIGN KEY(business_id) REFERENCES business(business_id)
);
INSERT INTO checkins(business_id, checkin_date)
SELECT parse_json($1):business_id, parse_json($1):date
FROM "UDACITYPROJECT"."STAGING".chekin;
-- tips
CREATE TABLE tips(
Results
Chart
number of rows inserted
131930
Query Details
Query duration
2.8s
Rows
1
Query ID
01bb8762-3201-873a-0...
Show more


climate and yelp integration
reporting and analysis
Staging to ODS
Databases
Worksheets
ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
UDACITYPROJECT.ODS
Settings
Code Versions
GAURAVADMIN
SNOWFLAKE
SNOWFLAKE_SAMPLE_DATA
UDACITYPROJECT
DWH
INFORMATION_SCHEMA
ODS
Tables
BUSINESS
REVIEWS
USER_TABLE
PUBLIC
STAGING
INSERT INTO checkins(business_id, checkin_date)
SELECT parse_json($1):business_id, parse_json($1):date
FROM "UDACITYPROJECT"."STAGING".chekin;
-- tips
CREATE TABLE tips(
tip_text VARCHAR,
tip_date VARCHAR(30),
compliment_count INT,
business_id VARCHAR(50),
user_id VARCHAR(22),
FOREIGN KEY(business_id) REFERENCES business(business_id),
FOREIGN KEY(user_id) REFERENCES user_table(user_id)
);
INSERT INTO tips(business_id, compliment_count, tip_date, tip_text, user_id)
SELECT parse_json($1):business_id, parse_json($1):compliment_count, parse_json($1):date, parse_json($1):text,
parse_json($1):user_id
FROM "UDACITYPROJECT"."STAGING".tip;
-- covid_feature
Results
Chart
number of rows inserted
908915
Query Details
Query duration
4.5s
Rows
1
Query ID
01bb8763-3201-873a-0...
Show more


climate and yelp integration
reporting and analysis
Staging to ODS
Databases
Worksheets
ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
UDACITYPROJECT.ODS
Settings
Code Versions
GAURAVADMIN
SNOWFLAKE
SNOWFLAKE_SAMPLE_DATA
UDACITYPROJECT
DWH
INFORMATION_SCHEMA
ODS
Tables
BUSINESS
CHECKINS
COVID_FEATURE
REVIEWS
TIPS
USER_TABLE
PUBLIC
STAGING
-- covid_feature
CREATE TABLE covid_feature(
business_id VARCHAR(50),
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(business_id)
);
INSERT INTO covid_feature(Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a_Quote_Enabled,
Temporary_Closed_Until,
Virtual_Services_Offered,business_id,delivery_or_takeout,highlights)
SELECT parse_json($1):"Call To Action enabled", parse_json($1):"Covid Banner", parse_json($1):"Grubhub enabled",
parse_json($1):"Request a Quote Enabled", parse_json($1):"Temporary Closed Until", parse_json($1):"Virtual Services
Offered", parse_json($1):"business_id", parse_json($1):"delivery or takeout", parse_json($1):"highlights"
FROM "UDACITYPROJECT"."STAGING".covid_features;
Results
Chart
number of rows inserted
209795
Query Details
Query duration
4.0s
Rows
1
Query ID
01bb8763-3201-87aa-0...
Show more



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

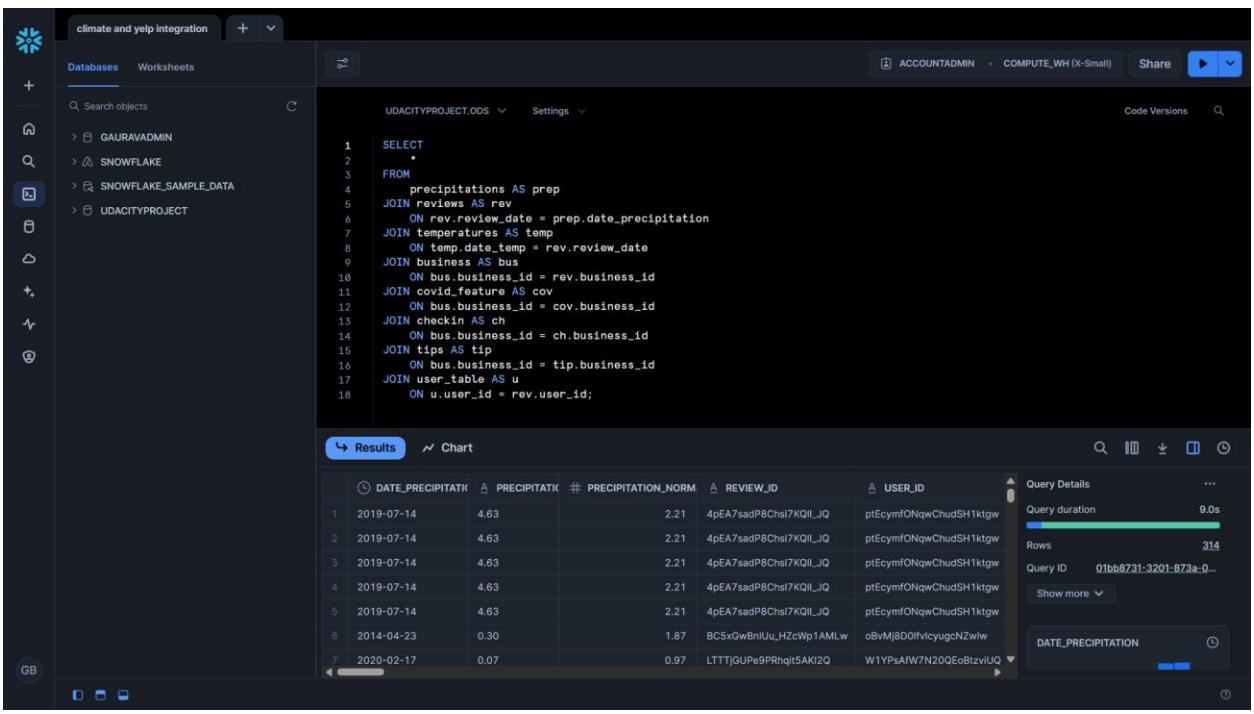
NAME_OF_TABLE	RAW	STAGING	ODS
CHECKIN	273.0 MB	80.5 MB	80.5 MB
COVIDS	61.8 MB	5.0 MB	5.0 MB
TIPS	172 MB	46.1 MB	46.9 MB
TEMPERATURE	800.0 KB	173.0 KB	173.0 KB
BUSINESS	113.0 MB	11.0 MB	11.9 MB
PRECIPITAION	516.0 KB	108.5 KB	108.5 KB
REVIEW	4.97 GB	1.9 GB	1.9 GB
USERS	3.13 GB	1.8 GB	1.8 GB

7. SQL queries that integrate the climate and Yelp datasets

```
SELECT
    *
FROM
    precipitations AS prep
JOIN reviews AS rev
    ON rev.review_date = prep.date_precipitation
JOIN temperatures AS temp
```

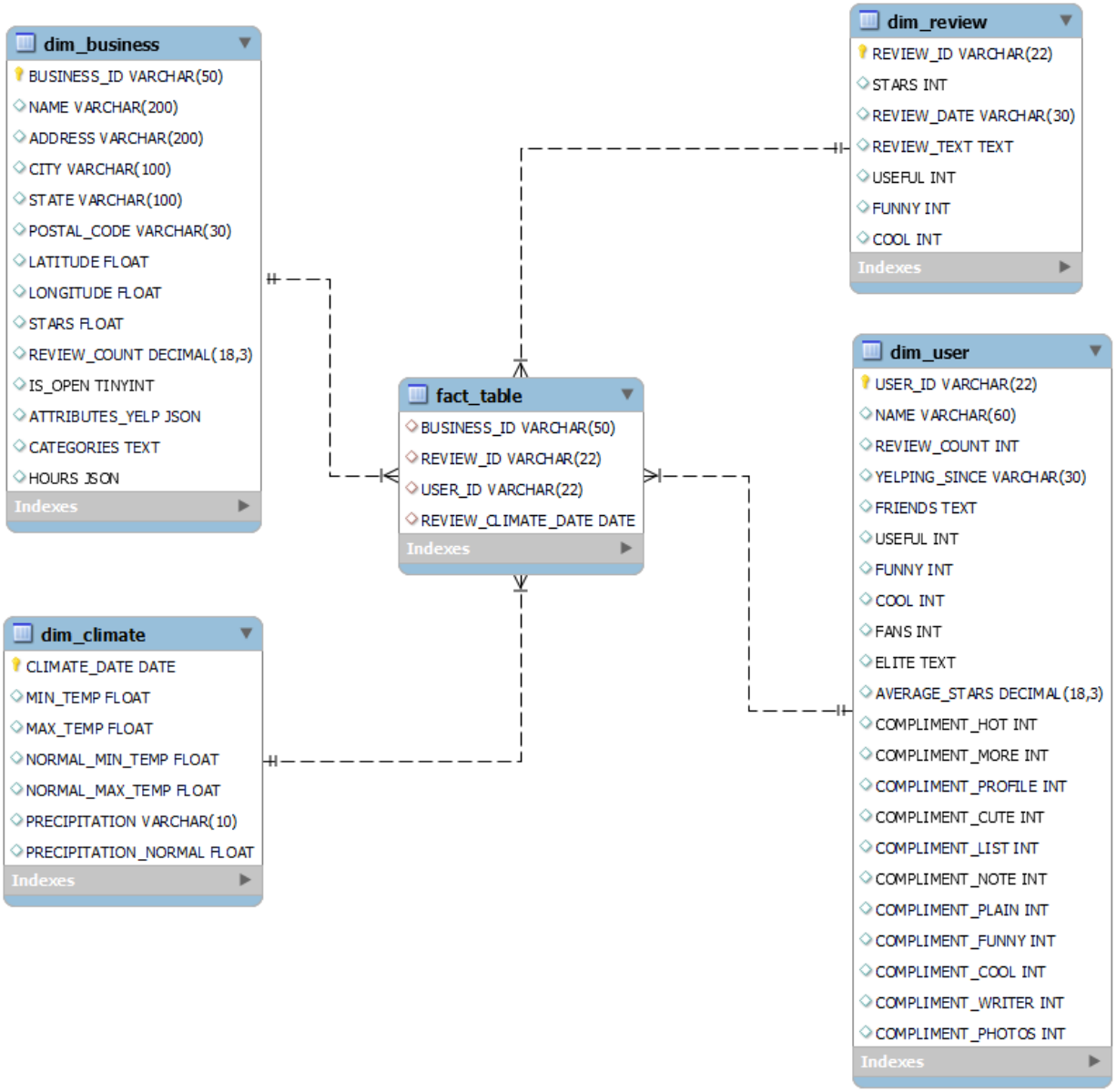
```
    ON temp.date_temp = rev.review_date
JOIN business AS bus
    ON bus.business_id = rev.business_id
JOIN covid_feature AS cov
    ON bus.business_id = cov.business_id
JOIN checkin AS ch
    ON bus.business_id = ch.business_id
JOIN tips AS tip
    ON bus.business_id = tip.business_id
JOIN user_table AS u
    ON u.user_id = rev.user_id;
```

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



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

```
CREATE TABLE DIM_BUSINESS(
   business_id VARCHAR(50) PRIMARY KEY,
   name VARCHAR(200),
```

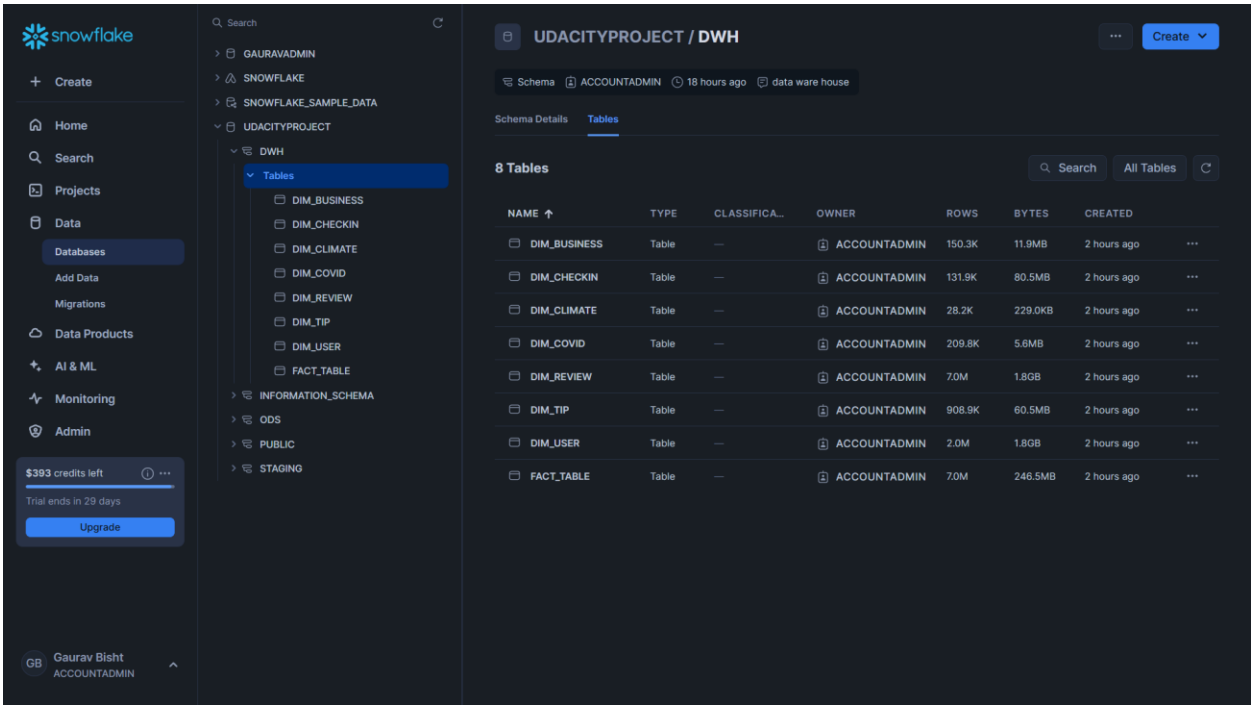
```sql
   address VARCHAR(200),
   city VARCHAR(100),
   state VARCHAR(100),
   postal_code VARCHAR(30),
   latitude FLOAT,
   longitude FLOAT,
   stars FLOAT,
   review_count DECIMAL(18,3),
   is_open INT,
   attribute_yelp VARIANT,
   categories VARCHAR,
   hours VARIANT
);
INSERT INTO DIM_BUSINESS(business_id, name,  address,  city, state,
postal_code, latitude,  longitude,  stars, review_count,
is_open, attribute_yelp, categories, hours)
SELECT business_id, name,  address,  city, state, postal_code, latitude,
longitude, stars, review_count,  is_open, attribute_yelp, categories, hours
FROM "UDACITYPROJECT"."ODS".business;

CREATE TABLE DIM_REVIEW(
    review_id VARCHAR(22) PRIMARY KEY,
    stars INT,
    review_date VARCHAR(30),
    review_text VARCHAR,
    useful INT,
    funny INT,
    cool INT
);
INSERT INTO DIM_REVIEW(cool,review_date,funny,review_id,stars,review_text,
useful)
SELECT cool,review_date,funny,review_id,stars,review_text, useful
FROM "UDACITYPROJECT"."ODS".reviews;

CREATE TABLE DIM_USER(
    user_id VARCHAR(22) PRIMARY KEY,
    name VARCHAR(60),
    review_count INT,
    yelping_since VARCHAR(30),
    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
DIM_USER(average_stars,compliment_cool,compliment_cute,compliment_funny,com
pliment_hot,compliment_list,
compliment_more,compliment_note,compliment_photos,compliment_plain,complime
nt_profile,compliment_writer, cool,elite, fans,friends,
funny,name,review_count, useful,user_id,yelping_since)
SELECT
average_stars,compliment_cool,compliment_cute,compliment_funny,compliment_h
ot,compliment_list,compliment_more,compliment_note,
compliment_photos,compliment_plain,compliment_profile,compliment_writer,coo
l,elite, fans,friends,funny,name,review_count,useful,user_id, yelping_since
FROM "UDACITYPROJECT"."ODS".user_table;

CREATE TABLE DIM_CHECKIN(
    business_id VARCHAR(22),
    checkin_date VARCHAR
);
INSERT INTO DIM_CHECKIN(business_id, checkin_date)
SELECT business_id, checkin_date
FROM "UDACITYPROJECT"."ODS".checkins;

CREATE TABLE DIM_TIP(
   tip_text VARCHAR,
   tip_date VARCHAR(30) PRIMARY KEY,
   compliment_count INT,
   business_id VARCHAR(22),
   user_id VARCHAR(22)
);
```

```
INSERT INTO DIM_TIP(business_id, compliment_count, tip_date, tip_text,
user_id)
SELECT business_id, compliment_count, tip_date, tip_text, user_id
FROM "UDACITYPROJECT"."ODS".tips;

CREATE TABLE DIM_COVID(
    business_id VARCHAR(22),
    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
DIM_COVID(Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a_Quo
te_Enabled,
Temporary_Closed_Until,Virtual_Services_Offered,business_id,delivery_or_tak
eout,highlights)
SELECT
Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a_Quote_Enabled
, Temporary_Closed_Until,
Virtual_Services_Offered,business_id,delivery_or_takeout,highlights
FROM "UDACITYPROJECT"."ODS".covid_feature;

CREATE TABLE DIM_CLIMATE(
   climate_date DATE PRIMARY KEY,
   min_temp FLOAT,
   max_temp FLOAT,
   normal_min_temp FLOAT,
   normal_max_temp FLOAT,
   precipitation VARCHAR(10),
   precipitation_normal FLOAT
);

INSERT INTO DIM_CLIMATE (climate_date, min_temp, max_temp,
normal_min_temp,normal_max_temp,precipitation,
precipitation_normal)
SELECT temp.date_temp, temp.min_temp, temp.max_temp,
temp.normal_min_temp,temp.normal_max_temp,prep.precipitation,
prep.precipitation_normal
```

```sql
FROM "UDACITYPROJECT"."ODS".temperatures AS temp
JOIN "UDACITYPROJECT"."ODS".precipitations AS prep
ON temp.DATE_TEMP = prep.DATE_PRECIPITATION;

create table FACT_TABLE(
   business_id VARCHAR(50),
   review_id VARCHAR(22),
   user_id VARCHAR(22),
   climate_date DATE,
   FOREIGN KEY (business_id) REFERENCES dim_business(business_id),
   FOREIGN KEY (review_id) REFERENCES dim_review(review_id),
   FOREIGN KEY (user_id) REFERENCES dim_user(user_id),
   FOREIGN KEY (climate_date) REFERENCES dim_climate(climate_date)
);
INSERT INTO fact_table(business_id,user_id,review_id,climate_date  )
SELECT rev.review_id,usr.user_id, bus.business_id, rev.review_date
FROM "UDACITYPROJECT"."ODS".reviews AS rev
JOIN "UDACITYPROJECT"."ODS".business AS bus
ON bus.business_id  = rev.business_id
JOIN "UDACITYPROJECT"."ODS".user_table AS usr
ON rev.user_id = usr.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

```sql
SELECT
    DISTINCT bus.name AS business_name,
    climate.max_temp,
    climate.min_temp,
    climate.precipitation,
    bus.stars as rating_stars
FROM FACT_TABLE AS fact
JOIN DIM_BUSINESS AS bus
    ON fact.business_id = bus.business_id
JOIN DIM_CLIMATE AS climate
    ON fact.review_climate_date  = climate.climate_date;
```

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