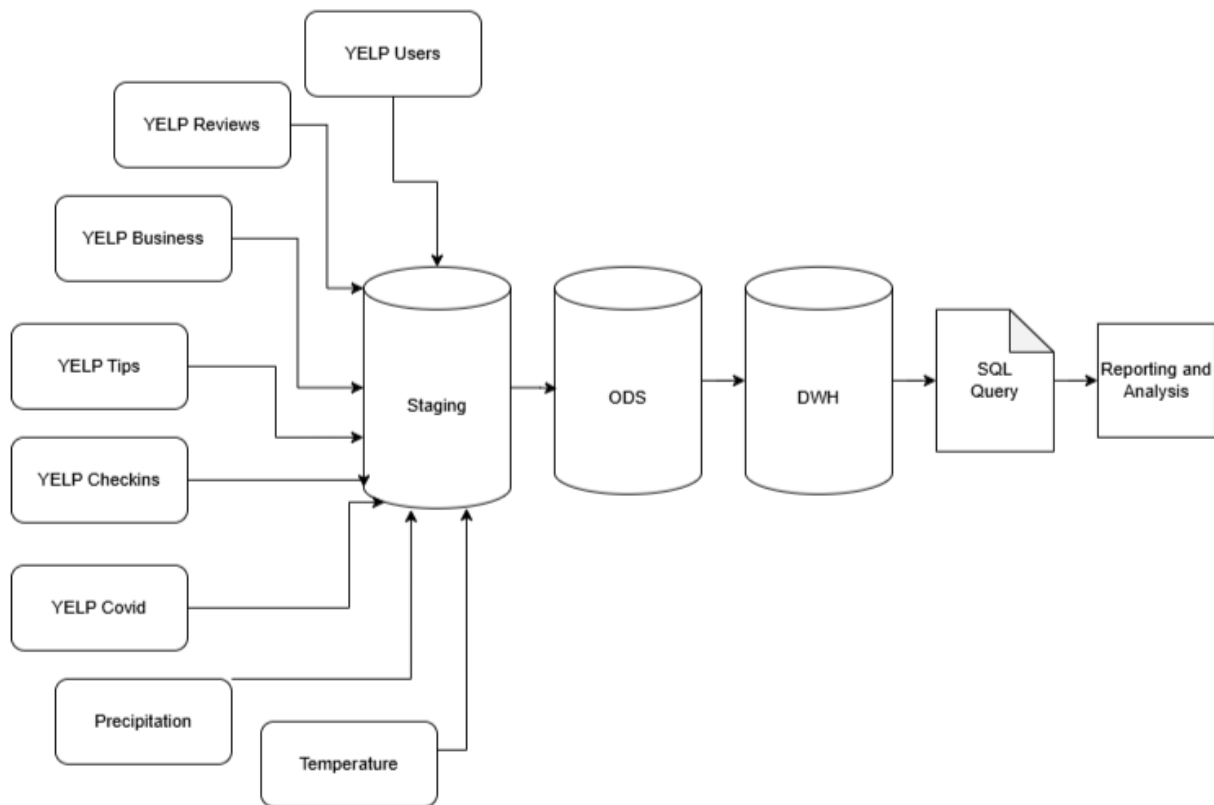


Submission Document - Design a Data Warehouse for Reporting and OLAP

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.

The screenshot shows the Snowflake web interface. On the left, the navigation sidebar is visible with options like Home, Search, Projects, Data, Databases, Add Data, Migrations, Data Products, AI & ML, Monitoring, and Admin. The main panel displays the 'UDACITYPROJECT / STAGING' schema. Under the 'Tables' tab, there are 6 tables listed:

NAME	TYPE	CLASSIFICA...	OWNER	ROWS	BYTES	CREATED
BUSINESS	Table	—	ACCOUNTADMIN	150.3K	11.0MB	15 hours a...
CHEKIN	Table	—	ACCOUNTADMIN	131.9K	80.5MB	15 hours a...
COVID_FEATURES	Table	—	ACCOUNTADMIN	209.8K	5.0MB	15 hours a...
REVIEW	Table	—	ACCOUNTADMIN	7.0M	1.9GB	15 hours a...
TIP	Table	—	ACCOUNTADMIN	908.9K	46.1MB	15 hours a...
USER_TABLE	Table	—	ACCOUNTADMIN	2.0M	1.8GB	15 hours a...

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

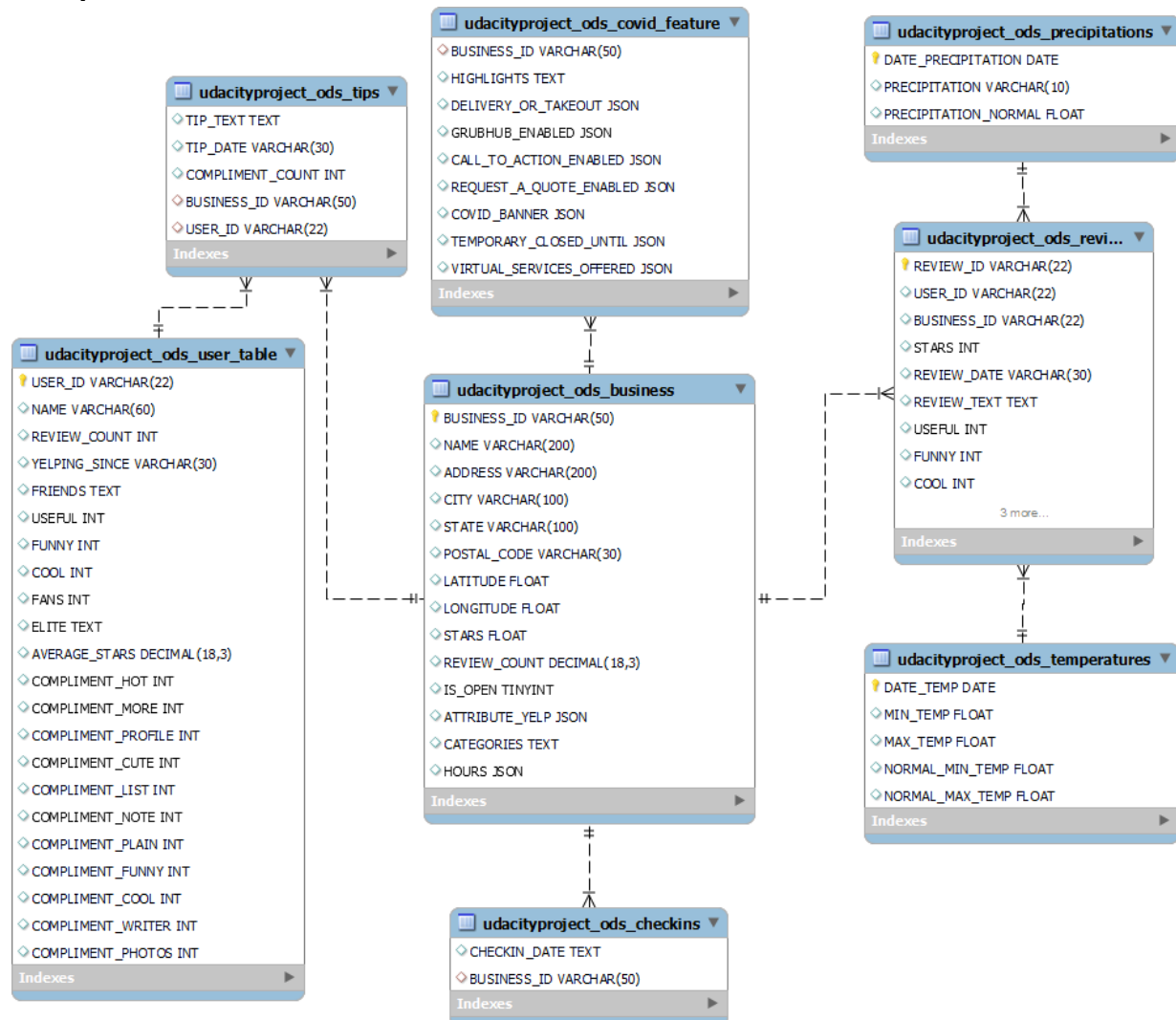
The screenshot should have two tables - temperature and precipitation.

The screenshot shows the Snowflake web interface. On the left, the navigation sidebar is visible with options like Home, Search, Projects, Data, Databases, Add Data, Migrations, Data Products, AI & ML, Monitoring, and Admin. The main panel displays the 'UDACITYPROJECT / ODS' schema. Under the 'Tables' tab, there are 8 tables listed:

NAME	TYPE	CLASSIFIC...	OWNER	ROWS	BYTES	CREATED
BUSINESS	Table	—	ACCOUNTADMIN	150.3K	11.9MB	1 hour ago
CHECKIN	Table	—	ACCOUNTADMIN	131.9K	80.5MB	1 hour ago
COVID_FEATURE	Table	—	ACCOUNTADMIN	209.8K	5.0MB	34 minutes...
PRECIPITATIONS	Table	—	ACCOUNTADMIN	28.2K	71.0KB	58 minutes...
REVIEWS	Table	—	ACCOUNTADMIN	7.0M	1.9GB	32 minutes...
TEMPERATURES	Table	—	ACCOUNTADMIN	28.2K	135.0KB	58 minutes...
TIPS	Table	—	ACCOUNTADMIN	908.9K	47.0MB	1 hour ago
USER_TABLE	Table	—	ACCOUNTADMIN	2.0M	1.9GB	1 hour ago

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
```

```
);  
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

The screenshot shows a Snowflake SQL editor interface. The left sidebar displays a database schema with the following structure:

- UDACITYPROJECT
 - DWH
 - INFORMATION_SCHEMA
 - ODS
 - Tables
 - BUSINESS
 - CHECKINS
 - COVID_FEATURE
 - PRECIPITATIONS
 - REVIEWS
 - TEMPERATURES
 - TIPS
 - USER_TABLE
 - PUBLIC
 - STAGING

The main editor displays the following SQL queries:

```
---
118 -- precipitations
119 CREATE TABLE precipitations(
120     date_precipitation DATE PRIMARY KEY,
121     precipitation VARCHAR(10),
122     precipitation_normal FLOAT
123 );
124 INSERT INTO precipitations(date_precipitation, precipitation, precipitation_normal)
125 SELECT TO_DATE(date_p), precipitation, precipitation_normal
126 FROM "UDACITYPROJECT"."STAGING".precipitation;
127
128 -- temperatures
129 CREATE TABLE temperatures(
130     date_temp DATE PRIMARY KEY,
131     min_temp FLOAT,
132     max_temp FLOAT,
133     normal_min_temp FLOAT,
134     normal_max_temp FLOAT
135 );
136 INSERT INTO TEMPERATURES(DATE_TEMP, MIN_TEMP, MAX_TEMP, NORMAL_MIN_TEMP, NORMAL_MAX_TEMP)
137 SELECT TO_DATE(DATE_T), MIN_TEMP, MAX_TEMP, NORMAL_MIN_TEMP, NORMAL_MAX_TEMP
138 FROM "UDACITYPROJECT"."STAGING".temperature;
139
```

The bottom panel shows the query results:

#	number of rows inserted
1	28241

Query Details:

- Query duration: 490ms
- Rows: 1
- Query ID: 01bb8765-3201-87aa-0...

The screenshot shows a Snowflake SQL editor interface. The left sidebar displays a database schema with the following structure:

- UDACITYPROJECT
 - DWH
 - INFORMATION_SCHEMA
 - ODS
 - Tables
 - BUSINESS
 - CHECKINS
 - COVID_FEATURE
 - REVIEWS
 - TIPS
 - USER_TABLE
 - PUBLIC
 - STAGING

The main editor displays the following SQL queries:

```
112 INSERT INTO covid_feature(Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a_Quote_Enabled,
113 Temporary_Closed_Until,
114 Virtual_Services_Offered,business_id,delivery_or_takeout,highlights)
115 SELECT parse_json($1):"Call To Action enabled", parse_json($1):"Covid Banner", parse_json($1):"Grubhub enabled",
116 parse_json($1):"Request a Quote Enabled", parse_json($1):"Temporary Closed Until", parse_json($1):"Virtual Services
117 Offered", parse_json($1):"business_id", parse_json($1):"delivery or takeout", parse_json($1):"highlights"
118 FROM "UDACITYPROJECT"."STAGING".covid_features;
119
120 -- precipitations
121 CREATE TABLE precipitations(
122     date_precipitation DATE PRIMARY KEY,
123     precipitation VARCHAR(10),
124     precipitation_normal FLOAT
125 );
126 INSERT INTO precipitations(date_precipitation, precipitation, precipitation_normal)
127 SELECT TO_DATE(date_p), precipitation, precipitation_normal
128 FROM "UDACITYPROJECT"."STAGING".precipitation;
129
130 -- temperatures
131 CREATE TABLE temperatures(
132     date_temp DATE PRIMARY KEY,
```

The bottom panel shows the query results:

#	number of rows inserted
1	28241

Query Details:

- Query duration: 466ms
- Rows: 1
- Query ID: 01bb8764-3201-873a-0...

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,
```

```

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,

```

```

        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".checkin;

-- 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

The screenshot shows a SQL query editor interface with a sidebar on the left containing a tree view of databases and worksheets. The main editor displays a SQL query for creating a table and inserting data from a JSON structure. The query is as follows:

```
1 -- business
2 CREATE TABLE business(
3     business_id VARCHAR(50) PRIMARY KEY,
4     name VARCHAR(200),
5     address VARCHAR(200),
6     city VARCHAR(100),
7     state VARCHAR(100),
8     postal_code VARCHAR(50),
9     latitude FLOAT,
10    longitude FLOAT,
11    stars FLOAT,
12    review_count DECIMAL(18,3),
13    is_open INT,
14    attribute_yelp VARIANT,
15    categories VARCHAR,
16    hours VARIANT
17 );
18 INSERT INTO business(business_id, name, address, city, state, postal_code, latitude, longitude, stars, review_count,
19 is_open, attribute_yelp, categories, hours)
20 SELECT parse_json($1):business_id, parse_json($1):name, parse_json($1):address, parse_json($1):city, parse_json($1):state,
21 parse_json($1):postal_code, parse_json($1):latitude, parse_json($1):longitude, parse_json($1):stars,
22 parse_json($1):review_count, parse_json($1):is_open, parse_json($1):attributes, parse_json($1):categories,
23 parse_json($1):hours
24 FROM "UDACITYPROJECT"."STAGING".business;
```

The results pane at the bottom shows a table with one row and one column, indicating the number of rows inserted. The query details pane on the right shows the query duration as 8.6s and the number of rows as 1.

#	number of rows inserted
1	150346

Query Details

- Query duration: 8.6s
- Rows: 1
- Query ID: 01bb875f-3201-873a-0...

The screenshot shows a SQL query editor interface with a sidebar on the left containing a tree view of databases and worksheets. The main editor displays a SQL query for creating a table and inserting data from a JSON structure. The query is as follows:

```
37 compliment_hot INT,
38 compliment_more INT,
39 compliment_profile INT,
40 compliment_cute INT,
41 compliment_list INT,
42 compliment_note INT,
43 compliment_plain INT,
44 compliment_funny INT,
45 compliment_cool INT,
46 compliment_writer INT,
47 compliment_photos INT
48 );
49 INSERT INTO user_table(average_stars, compliment_cool, compliment_cute, compliment_funny, compliment_hot, compliment_list, compliment_more, compliment_note,
50 compliment_photos, compliment_plain, compliment_profile, compliment_writer, cool, elite,
51 fans, friends, funny, name, review_count, useful, user_id, yelping_since)
52 SELECT parse_json($1):average_stars, parse_json($1):compliment_cool, parse_json($1):compliment_cute, parse_json($1):compliment_funny,
53 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,
54 parse_json($1):compliment_plain, parse_json($1):compliment_profile, parse_json($1):compliment_writer, parse_json($1):cool, parse_json($1):elite,
55 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,
56 parse_json($1):user_id, parse_json($1):yelping_since
57 FROM "UDACITYPROJECT"."STAGING".user_table;
```

The results pane at the bottom shows a table with one row and one column, indicating the number of rows inserted. The query details pane on the right shows the query duration as 20s and the number of rows as 1.

#	number of rows inserted
1	1987897

Query Details

- Query duration: 20s
- Rows: 1
- Query ID: 01bb875f-3201-873a-0...

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

```
52 parse_json($1):elite, parse_json($1):fans, parse_json($1):friends, parse_json($1):funny, parse_json($1):name,
53 parse_json($1):review_count, parse_json($1):useful, parse_json($1):user_id, parse_json($1):yelping_since
54 FROM "UDACITYPROJECT"."STAGING".user_table;
55
56 -- reviews
57 CREATE TABLE reviews(
58   review_id VARCHAR(22) PRIMARY KEY,
59   user_id VARCHAR(22),
60   business_id VARCHAR(22),
61   stars INT,
62   review_date VARCHAR(30),
63   review_text VARCHAR,
64   useful INT,
65   funny INT,
66   cool INT
67 );
68 INSERT INTO reviews(business_id, cool, review_date, funny, review_id, stars, review_text, useful, user_id)
69 SELECT parse_json($1):business_id, parse_json($1):cool, parse_json($1):date, parse_json($1):funny, parse_json($1):review_id,
70 parse_json($1):stars, parse_json($1):text, parse_json($1):useful, parse_json($1):user_id
71 FROM "UDACITYPROJECT"."STAGING".review;
```

Results Chart

#	number of rows inserted
1	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

```
65 cool INT
66 );
67 INSERT INTO reviews(business_id, cool, review_date, funny, review_id, stars, review_text, useful, user_id)
68 SELECT parse_json($1):business_id, parse_json($1):cool, parse_json($1):date, parse_json($1):funny, parse_json($1):review_id,
69 parse_json($1):stars, parse_json($1):text, parse_json($1):useful, parse_json($1):user_id
70 FROM "UDACITYPROJECT"."STAGING".cheekin;
71
72 -- checkins
73 CREATE TABLE checkins(
74   checkin_date VARCHAR,
75   business_id VARCHAR(50),
76   FOREIGN KEY(business_id) REFERENCES business(business_id)
77 );
78
79 INSERT INTO checkins(business_id, checkin_date)
80 SELECT parse_json($1):business_id, parse_json($1):date
81 FROM "UDACITYPROJECT"."STAGING".cheekin;
82
83 -- tips
84 CREATE TABLE tips(
85   tip_text VARCHAR
86 );
```

Results Chart

#	number of rows inserted
1	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

Search objects

- GAURAVADMIN
- SNOWFLAKE
- SNOWFLAKE_SAMPLE_DATA
- UDACITYPROJECT
 - DWH
 - INFORMATION_SCHEMA
 - ODS
 - Tables
 - BUSINESS
 - REVIEWS
 - USER_TABLE
 - PUBLIC
 - STAGING

```
--
79 INSERT INTO checkins(business_id, checkin_date)
80 SELECT parse_json($1):business_id, parse_json($1):date
81 FROM "UDACITYPROJECT"."STAGING".checkin;
82
83
84 -- tips
85 CREATE TABLE tips(
86   tip_text VARCHAR,
87   tip_date VARCHAR(30),
88   compliment_count INT,
89   business_id VARCHAR(50),
90   user_id VARCHAR(22),
91   FOREIGN KEY(business_id) REFERENCES business(business_id),
92   FOREIGN KEY(user_id) REFERENCES user_table(user_id)
93 );
94 INSERT INTO tips(business_id, compliment_count, tip_date, tip_text, user_id)
95 SELECT parse_json($1):business_id, parse_json($1):compliment_count, parse_json($1):date, parse_json($1):text,
96        parse_json($1):user_id
97 FROM "UDACITYPROJECT"."STAGING".tips;
98
99 -- covid.feature
100 CREATE TABLE covid_feature(
```

Results Chart

#	number of rows inserted
1	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

Search objects

- GAURAVADMIN
- SNOWFLAKE
- SNOWFLAKE_SAMPLE_DATA
- UDACITYPROJECT
 - DWH
 - INFORMATION_SCHEMA
 - ODS
 - Tables
 - BUSINESS
 - CHECKINS
 - COVID_FEATURE
 - REVIEWS
 - TIPS
 - USER_TABLE
 - PUBLIC
 - STAGING

```
97
98 -- covid.feature
99 CREATE TABLE covid_feature(
100   business_id VARCHAR(50),
101   highlights VARCHAR,
102   delivery_or_takeout VARIANT,
103   grubhub_enabled VARIANT,
104   call_to_action_enabled VARIANT,
105   request_a_quote_enabled VARIANT,
106   covid_banner VARIANT,
107   temporary_closed_until VARIANT,
108   virtual_services_offered VARIANT,
109   FOREIGN KEY (business_id) REFERENCES business(business_id)
110 );
111
112 INSERT INTO covid_feature(Call_To_Action_enabled,Covid_Banner,Grubhub_enabled,Request_a_Quote_Enabled,
113 Temporary_Closed_Until,
114 Virtual_Services_Offered,business_id,delivery_or_takeout,highlights)
115 SELECT parse_json($1):"Call To Action enabled", parse_json($1):"Covid Banner", parse_json($1):"Grubhub enabled",
116        parse_json($1):"Request a Quote Enabled", parse_json($1):"Temporary Closed Until", parse_json($1):"Virtual Services
117 Offered", parse_json($1):"business_id", parse_json($1):"delivery or takeout", parse_json($1):"highlights"
118 FROM "UDACITYPROJECT"."STAGING".covid_features;
119
```

Results Chart

#	number of rows inserted
1	209795

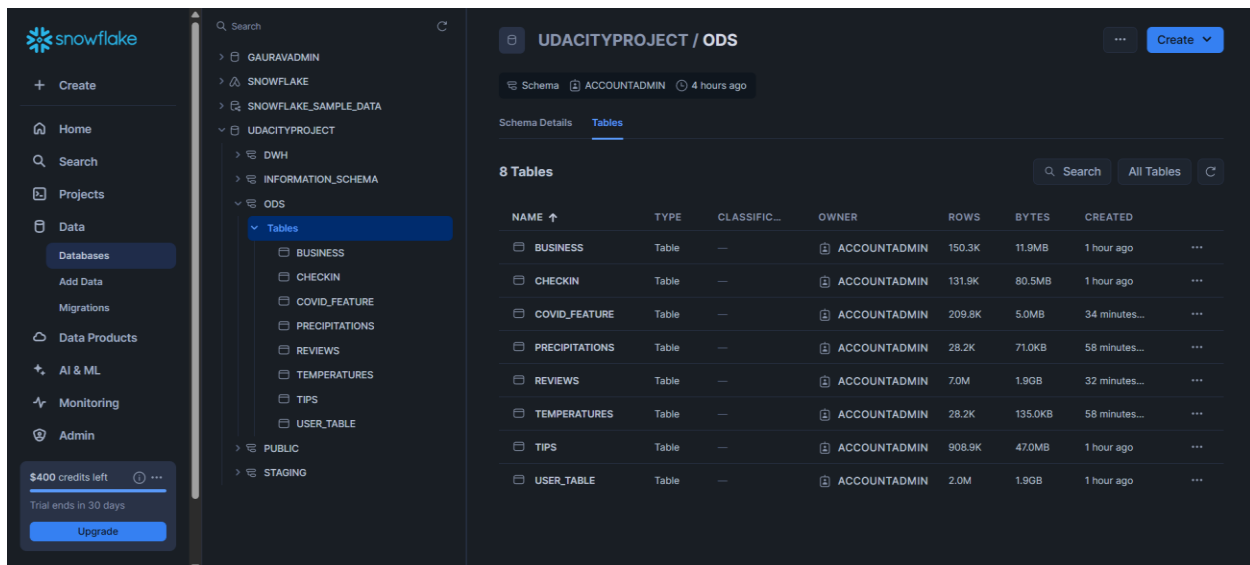
Query Details

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

The screenshot displays a Snowflake SQL query editor interface. The query is a complex JOIN statement integrating multiple datasets. The results pane shows a table with 7 rows and 5 columns: DATE_PRECIPITATK, PRECIPITATK, PRECIPITATION_NORM, REVIEW_ID, and USER_ID.

Query:

```

1  SELECT
2  *
3  FROM
4  precipitations AS prep
5  JOIN reviews AS rev
6  ON rev.review_date = prep.date_precipitation
7  JOIN temperatures AS temp
8  ON temp.date_temp = rev.review_date
9  JOIN business AS bus
10 ON bus.business_id = rev.business_id
11 JOIN covid_feature AS cov
12 ON bus.business_id = cov.business_id
13 JOIN checkin AS ch
14 ON bus.business_id = ch.business_id
15 JOIN tips AS tip
16 ON bus.business_id = tip.business_id
17 JOIN user_table AS u
18 ON u.user_id = rev.user_id;

```

Results Table:

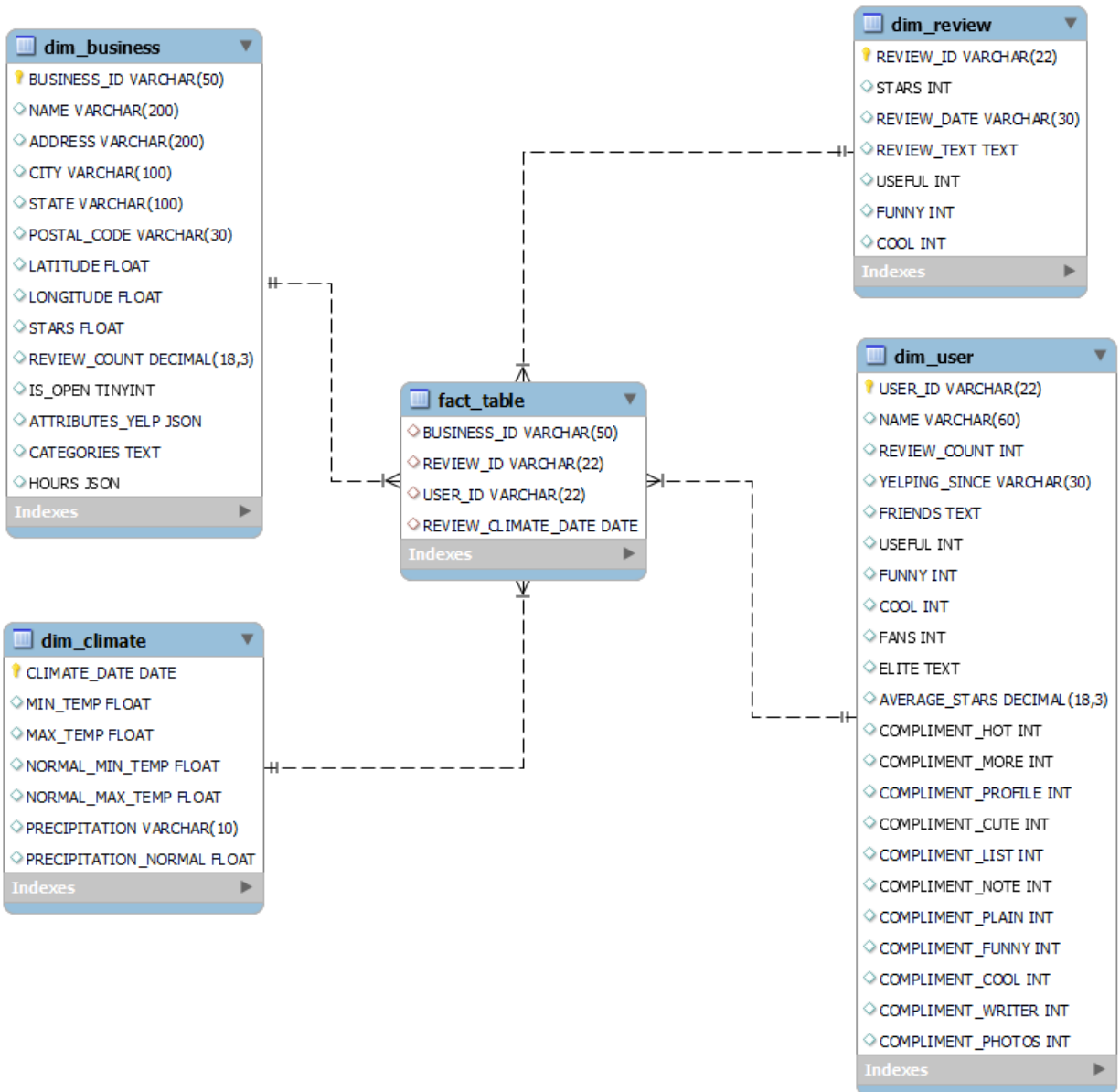
	DATE_PRECIPITATK	PRECIPITATK	PRECIPITATION_NORM	REVIEW_ID	USER_ID
1	2019-07-14	4.63	2.21	4pEA7sadP8Chs17KQIL_JQ	ptEcymfONqWChudSH1ktgw
2	2019-07-14	4.63	2.21	4pEA7sadP8Chs17KQIL_JQ	ptEcymfONqWChudSH1ktgw
3	2019-07-14	4.63	2.21	4pEA7sadP8Chs17KQIL_JQ	ptEcymfONqWChudSH1ktgw
4	2019-07-14	4.63	2.21	4pEA7sadP8Chs17KQIL_JQ	ptEcymfONqWChudSH1ktgw
5	2019-07-14	4.63	2.21	4pEA7sadP8Chs17KQIL_JQ	ptEcymfONqWChudSH1ktgw
6	2014-04-23	0.30	1.87	BCSxGwBnlUu_HZcWp1Amlw	oBvMj8D0IfvicygcNZwlv
7	2020-02-17	0.07	0.97	LTTTjOUpe9Prngit5AKI2Q	W1YPsAfW7N20Qe0BzviUQ

Query Details:

- Query duration: 9.0s
- Rows: 314
- Query ID: 01bb8731-3201-873a-0...

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),
```



```

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,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
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
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_Quote_Enabled,
Temporary_Closed_Until,Virtual_Services_Offered,business_id,delivery_or_takeout,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

```

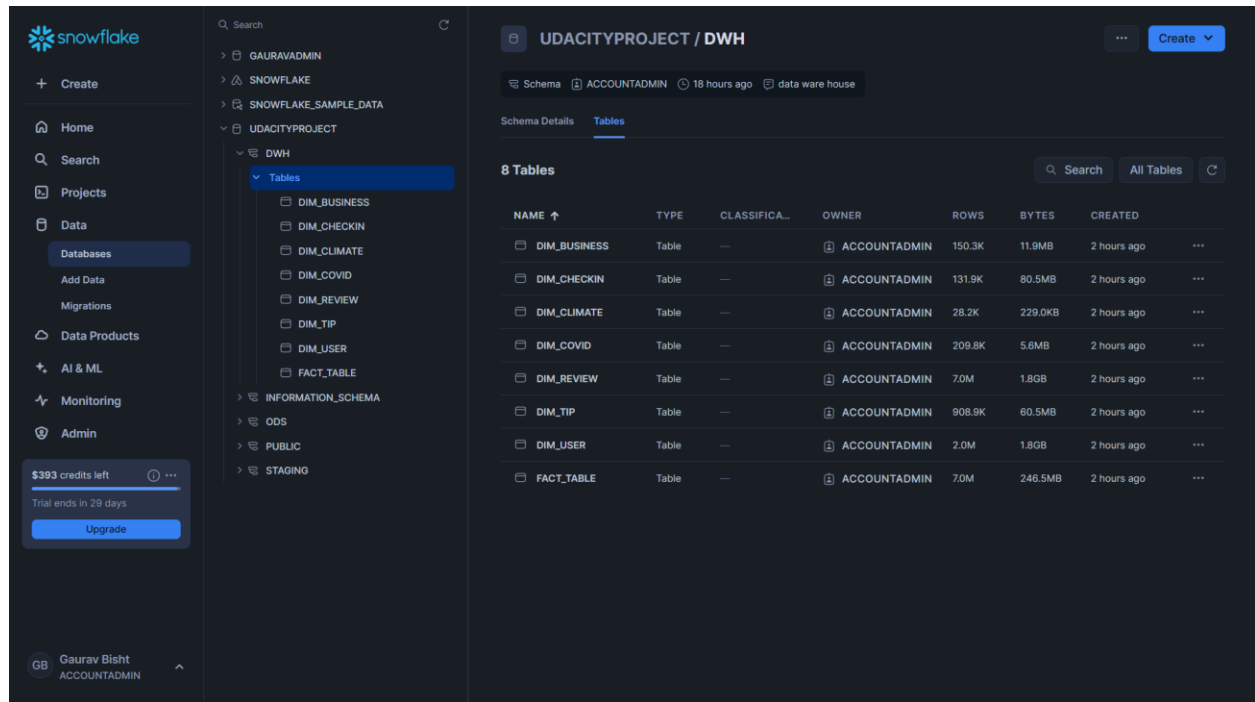
```

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



The screenshot shows the Snowflake web interface. On the left is a navigation sidebar with options like Home, Search, Projects, Data, Databases, Add Data, Migrations, Data Products, AI & ML, Monitoring, and Admin. The main panel displays the 'UDACITYPROJECT / DWH' schema. Under the 'Tables' tab, a list of 8 tables is shown:

NAME	TYPE	CLASSIFICA...	OWNER	ROWS	BYTES	CREATED
DIM_BUSINESS	Table	—	ACCOUNTADMIN	150.3K	11.9MB	2 hours ago
DIM_CHECKIN	Table	—	ACCOUNTADMIN	131.9K	80.5MB	2 hours ago
DIM_CLIMATE	Table	—	ACCOUNTADMIN	28.2K	229.0KB	2 hours ago
DIM_COVID	Table	—	ACCOUNTADMIN	209.8K	5.6MB	2 hours ago
DIM_REVIEW	Table	—	ACCOUNTADMIN	7.0M	1.8GB	2 hours ago
DIM_TIP	Table	—	ACCOUNTADMIN	908.9K	60.5MB	2 hours ago
DIM_USER	Table	—	ACCOUNTADMIN	2.0M	1.8GB	2 hours ago
FACT_TABLE	Table	—	ACCOUNTADMIN	7.0M	246.5MB	2 hours ago

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

```
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

The screenshot displays the Snowflake web interface. The top navigation bar includes tabs for 'climate and yelp integration' and 'reporting and analysis'. The left sidebar shows a tree view of databases and schemas, with 'UDACITYPROJECT' selected. The main panel shows a SQL query in the 'UDACITYPROJECT.DWH' database. The query is a SELECT statement that joins 'FACT_TABLE' with 'DIM_BUSINESS' and 'DIM_CLIMATE' to retrieve distinct business names, maximum and minimum temperatures, precipitation, and rating stars. The results are displayed in a table with 11 rows. A 'Query Details' panel on the right shows the query duration (1.4s), the number of rows (3,387), and the query ID. A 'Business Name' dropdown menu is also visible, showing a list of businesses and their counts.

```
1 SELECT
2   DISTINCT bus.name AS business_name,
3   climate.max_temp,
4   climate.min_temp,
5   climate.precipitation,
6   bus.stars AS rating_stars
7 FROM FACT_TABLE AS fact
8 JOIN DIM_BUSINESS AS bus
9   ON fact.business_id = bus.business_id
10 JOIN DIM_CLIMATE AS climate
11   ON fact.review_climate_date = climate.climate_date;;
```

	BUSINESS_NAME	MAX_TEMP	MIN_TEMP	PRECIPITATION	RATING_STARS
1	Kingston City Restaurant	66	91	3.33	2.5
2	Philly Fade Factory Barber	34	50	7	4
3	U.S. Defense Solutions	85	112	0.47	5
4	Wonder Nail	55	81	0.26	4
5	Accura Systems	88	108	3.29	4.5
6	Roux Public House	76	95	1.19	3
7	Fernando's Smog Check	79	107	0.87	5
8	Persnickety Stitchers	47	70	2.71	4.5
9	Subway	82	104	1.03	1.5
10	Dino's Backstage	68	90	1.51	3.5
11	Micro Movers	74	95	1.17	5

Query Details

- Query duration: 1.4s
- Rows: 3,387
- Query ID: 01bb8737-3201-87aa-0...

Business Name

- Starbucks: 18
- McDonald's: 13
- Subway: 10
- + 3,067 more