

# Acrotrend

From Zero to Snowflake In 90 minutes

Participant Guide

*Dikesh Shah*

1. Preface	3
2. Prerequisites	4
3. Materials and set-up	5
Files:	5
Snowflake:	5
4. What you'll learn and how you can apply it	6
5. Snowflake UI Navigation	7
Logging in to Snowflake	7
Navigating the Snowflake UI	8
6. Database Setup	11
7. Data Preparation for loading into Snowflake	14
Create SWIPE_DATA table.	14
Create an External Stage	15
Create a File Format	16
8. Loading data	20
Create warehouse	20
Change the context	22
Load the data	22
Validate the data loaded	27
9. Running Analytical Queries:	29
Get basic counts of gym members from swipe data:	29
Most busiest day of the week	29
Active gym members	30
Top 10 active gym members	31
10. Data Load and Errors	32
Create file stage, file format and table for data load	32
Load data and identify error	33
Error analysis and reload of data	33
11. Working with semi-structured files	36
JSON Sample data	36
Create JSON file format	37
Create JSON File Stage	37
Check the files present on the JSON Stage	37
Create a staging table with VARIANT datatype	38
Load the JSON files in the new table created	38
Verify the data load	39
Create view to query the JSON data into structured form	41
Get location details for swipe data	42

Insert the data into the existing SWIPE_DATA table	42
<b>12. DML Operations on data</b>	<b>43</b>
Create and Insert data in a new table (using create table as an option)	43
<b>13. Mistakes and how to recover data using undrop, time-travel and zero-cloning</b>	<b>46</b>
Drop and undrop table	46
Time-travel	47
Clone table	49
<b>14. Data Sharing</b>	<b>51</b>
Data Sharing Understanding	51
Create outbound share	51
Grant usage on CHURN_DATA_ANALYSIS database and schema to the share:	52
Create a new reader account and add it to the share.	52
Import the share from the Consumer account (eader Account)	54
<b>15. Appendix A</b>	<b>56</b>

## 1. Preface

This guide is the supplement to the workshop “From Zero to Snowflake In 90 Minutes”.

## 2. Prerequisites

This workshop assumes that you know the basics of SQL and database structures.

### 3. Materials and set-up

#### Files:

You should be able to access the csv and JSON files located at the AWS S3 bucket.

Note the bucket is a read-only bucket and everyone should have read-only access on the contents of the bucket.

File	AWS S3 Bucket Name
CSV	s3://acro-snowflake-workshop/CSV/
JSON	s3://acro-snowflake-workshop /JSON/
Location	S3://acro-snowflake-workshop/
Demo Script:	<a href="https://s3.eu-west-2.amazonaws.com/acro-snowflake-workshop/DEMO_SCRIPT.txt">https://s3.eu-west-2.amazonaws.com/acro-snowflake-workshop/DEMO_SCRIPT.txt</a>
Hands On Guide	<a href="https://s3.eu-west-2.amazonaws.com/acro-snowflake-workshop/Acrotrend Zero to Snowflake Participant Guide.pdf">https://s3.eu-west-2.amazonaws.com/acro-snowflake-workshop/Acrotrend Zero to Snowflake Participant Guide.pdf</a>

#### Snowflake:

You should have the Snowflake URL login with your username/password. If you don't have this then you can register for the Snowflake Trial Account.

## 4. What you'll learn and how you can apply it

- How to access Snowflake instance and navigate the UI.
- How to load CSV files and run analytical queries over them.
- How to work with semi-structured data and load them in Snowflake using SQL.
- How to use time-travel feature to recover data.
- How to use data sharing feature to share data for consumption.
- How to provide data security and limit access to data based on role type.

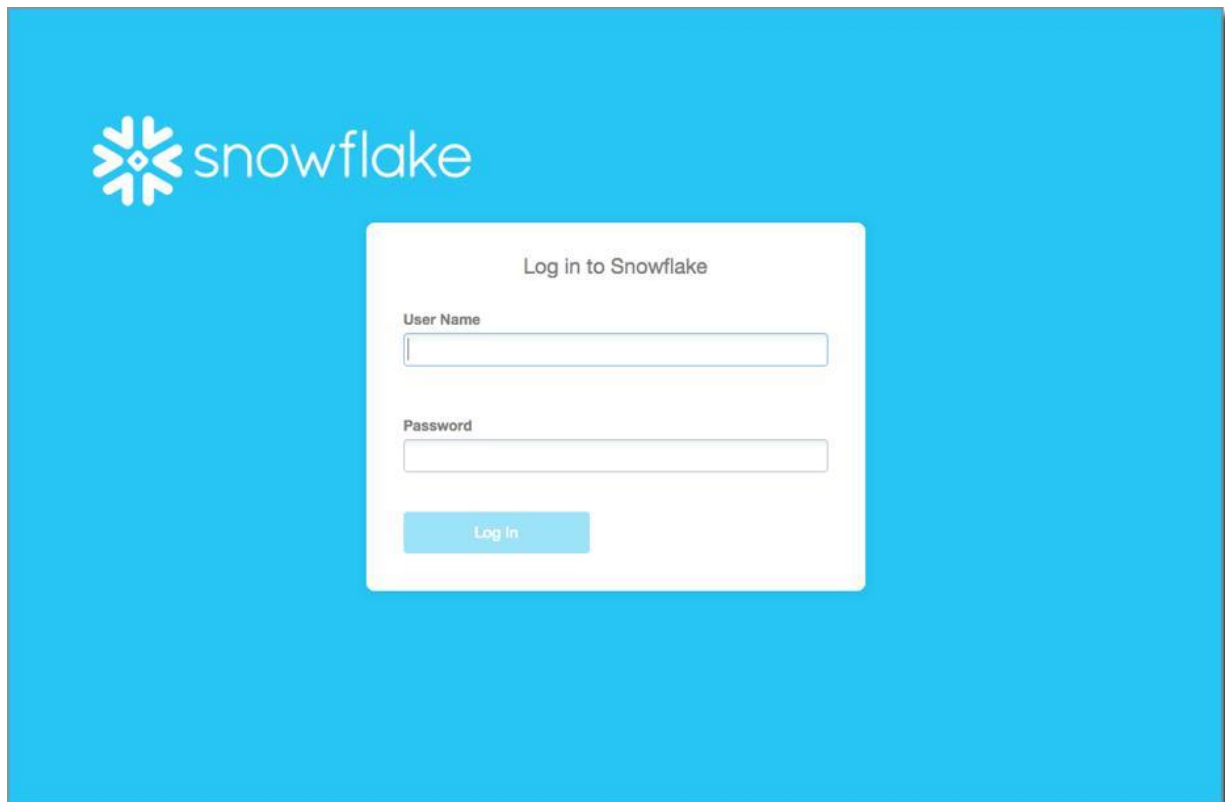
## 5. Snowflake UI Navigation

### Logging in to Snowflake

Open a browser window and launch the demo environment:

<https://<yoururlhere>.com>

You should see the following login screen:



Enter your credentials as below and click Log In:

User Name:

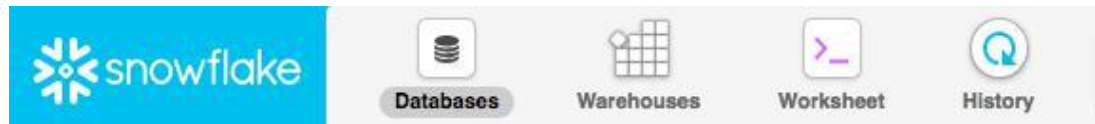
Password:



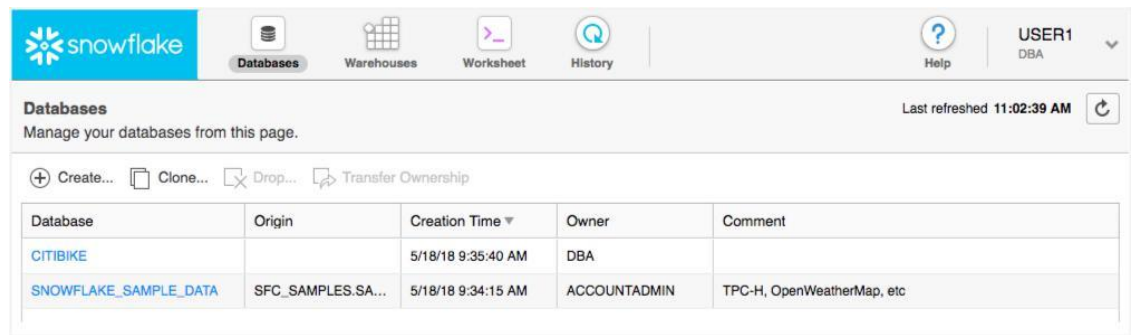
## Navigating the Snowflake UI

First let's get you acquainted with Snowflake! This section covers the basic components of the user interface to help you orient yourself.

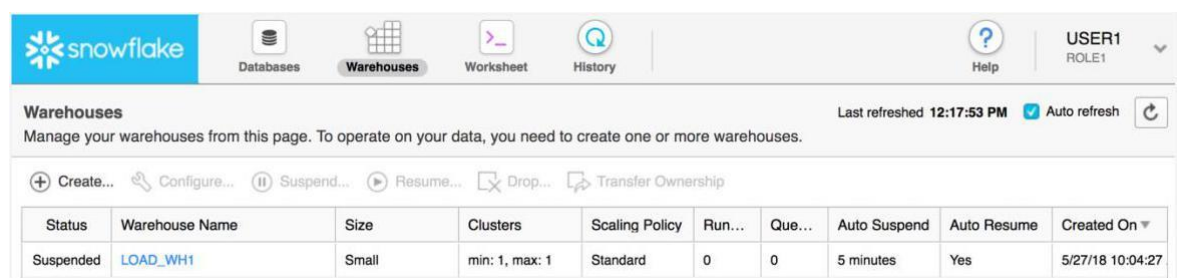
The top menu allows you to switch between the different areas of Snowflake:



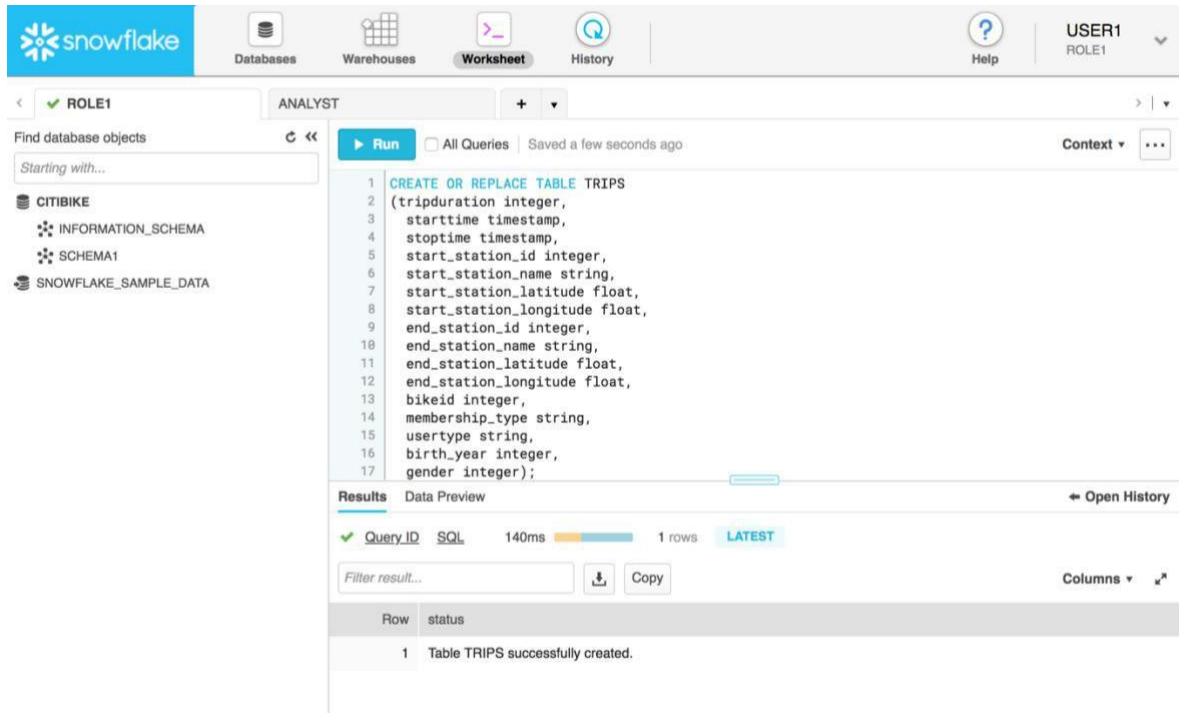
The Databases tab shows information about the databases you have created or have privileges to access. You can create, clone, drop, or transfer ownership of databases as well as load data (limited) in the UI:



The Warehouses tab is where you set up and manage compute resources (virtual warehouses) to load or query data in Snowflake:

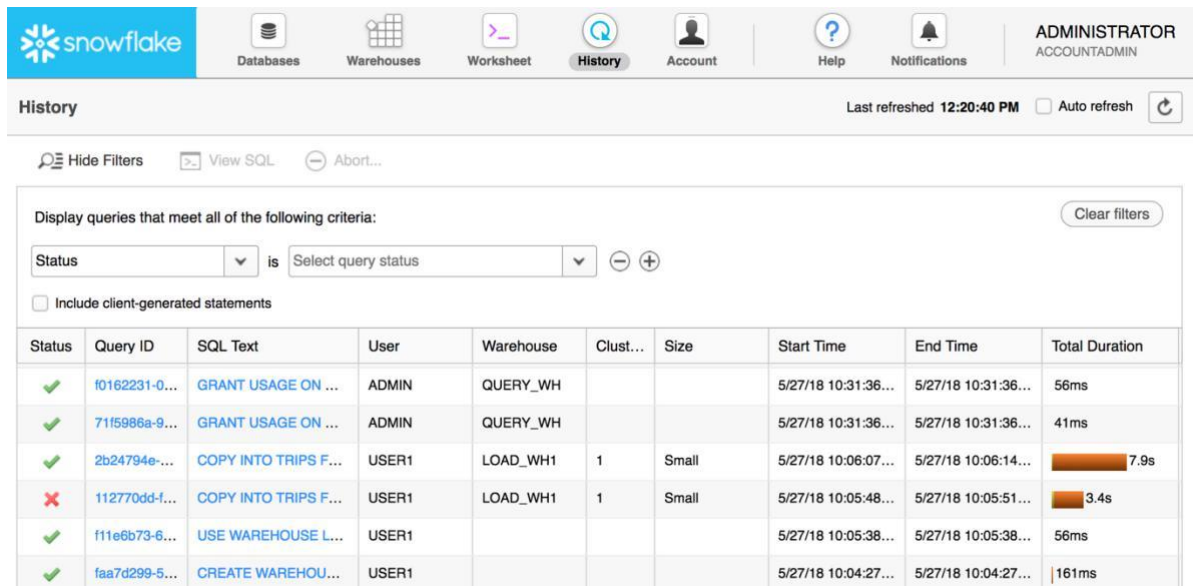


The Worksheet provides an interface for submitting SQL queries, performing DDL and DML operations and viewing results as your queries/operations complete:



The screenshot shows the Snowflake Worksheet interface. The top navigation bar includes tabs for Databases, Warehouses, Worksheet (active), and History. The user is logged in as USER1 (ROLE1). The left sidebar shows the database structure for CITIBIKE, including INFORMATION\_SCHEMA, SCHEMA1, and SNOWFLAKE\_SAMPLE\_DATA. The main area displays a SQL query to create a table named TRIPS. Below the query, the 'Results' section shows a data preview with 1 row: 'Table TRIPS successfully created.' The query execution took 140ms.

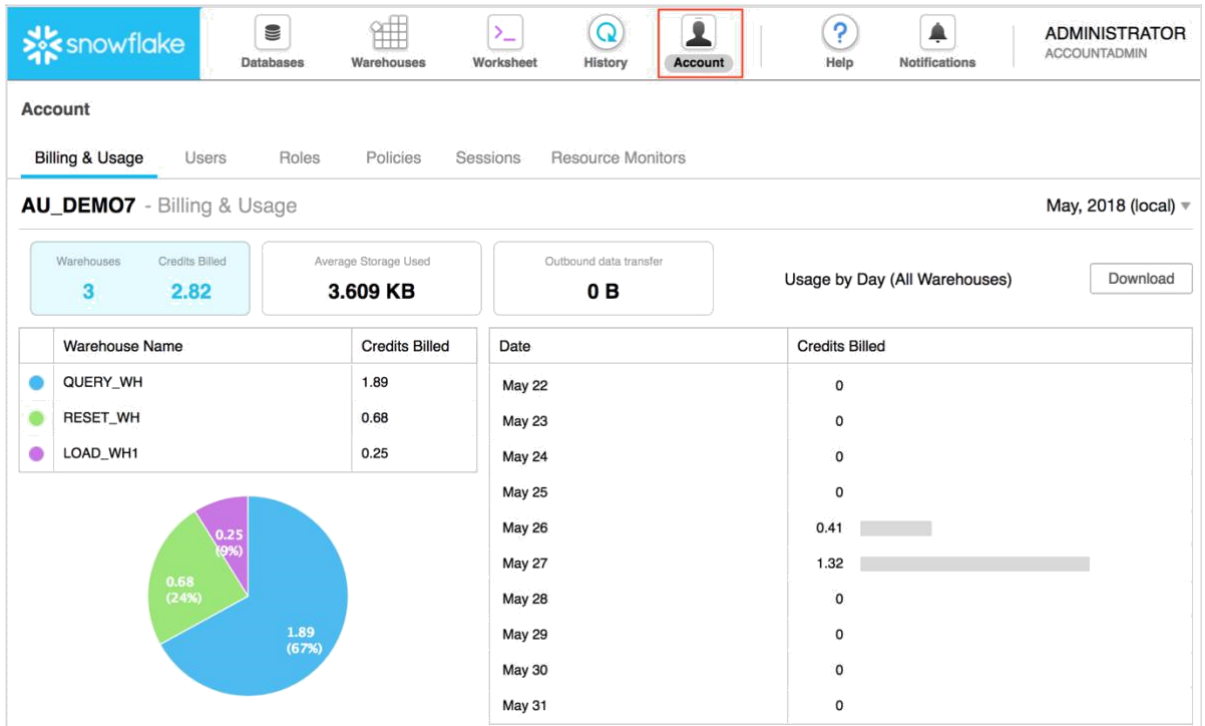
The History tab allows you to view the details of all queries executed in the last 14 days (click on a Query ID to drill down):



The screenshot shows the Snowflake History tab. The top navigation bar includes tabs for Databases, Warehouses, Worksheet, History (active), Account, and Help. The user is logged in as ADMINISTRATOR (ACCOUNTADMIN). The History section shows a list of queries executed in the last 14 days. The table below lists the queries with their status, query ID, SQL text, user, warehouse, cluster, size, start time, end time, and total duration.

Status	Query ID	SQL Text	User	Warehouse	Clust...	Size	Start Time	End Time	Total Duration
✓	10162231-0...	GRANT USAGE ON ...	ADMIN	QUERY_WH			5/27/18 10:31:36...	5/27/18 10:31:36...	56ms
✓	71f5986a-9...	GRANT USAGE ON ...	ADMIN	QUERY_WH			5/27/18 10:31:36...	5/27/18 10:31:36...	41ms
✓	2b24794e-...	COPY INTO TRIPS F...	USER1	LOAD_WH1	1	Small	5/27/18 10:06:07...	5/27/18 10:06:14...	7.9s
✗	112770dd-f...	COPY INTO TRIPS F...	USER1	LOAD_WH1	1	Small	5/27/18 10:05:48...	5/27/18 10:05:51...	3.4s
✓	f11e6b73-6...	USE WAREHOUSE L...	USER1				5/27/18 10:05:38...	5/27/18 10:05:38...	56ms
✓	faa7d299-5...	CREATE WAREHOU...	USER1				5/27/18 10:04:27...	5/27/18 10:04:27...	161ms

On switching to the ACCOUNTADMIN role, you will see a fifth tab called Account.



The screenshot displays the Snowflake Account management interface. The top navigation bar includes links for Databases, Warehouses, Worksheet, History, Account (highlighted), Help, and Notifications. The user is logged in as ADMINISTRATOR (ACCOUNTADMIN). The main section is titled 'Account' and includes tabs for Billing & Usage, Users, Roles, Policies, Sessions, and Resource Monitors. The 'Billing & Usage' tab is active, showing account details for 'AU\_DEMO7' as of May 2018. Key metrics include 3 Warehouses, 2.82 Credits Billed, 3.609 KB Average Storage Used, and 0 B Outbound data transfer. A 'Usage by Day (All Warehouses)' table and a corresponding bar chart are also visible. A pie chart shows the distribution of credits billed across three warehouses: QUERY\_WH (1.89, 67%), RESET\_WH (0.68, 24%), and LOAD\_WH1 (0.25, 9%).

Warehouse Name	Credits Billed
QUERY_WH	1.89
RESET_WH	0.68
LOAD_WH1	0.25

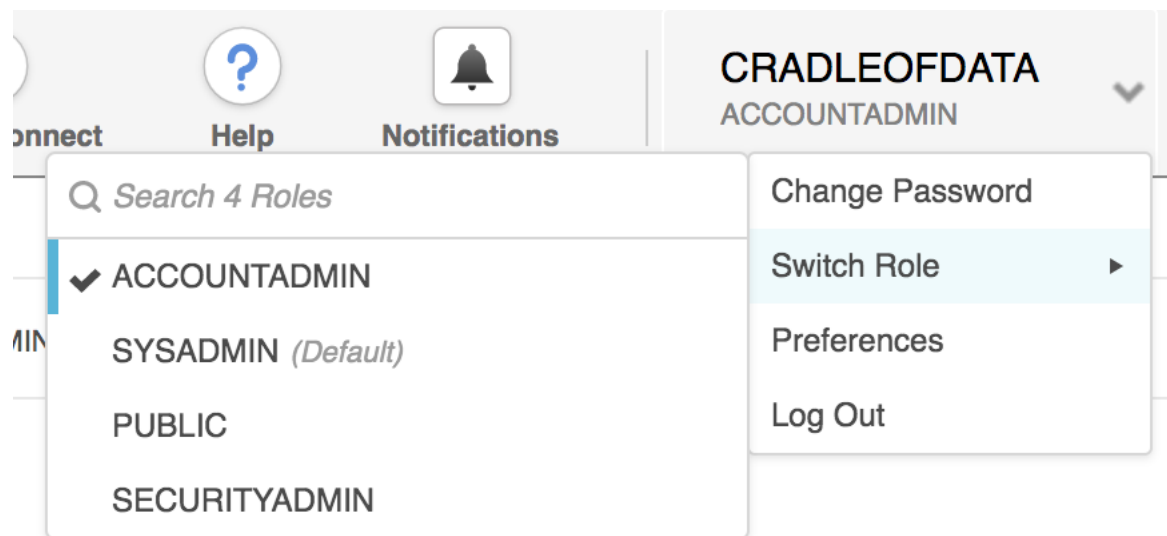
Date	Credits Billed
May 22	0
May 23	0
May 24	0
May 25	0
May 26	0.41
May 27	1.32
May 28	0
May 29	0
May 30	0
May 31	0

This is where you can add users/roles, configure network policies, set up resource monitors, and check your billing and usage statistics.

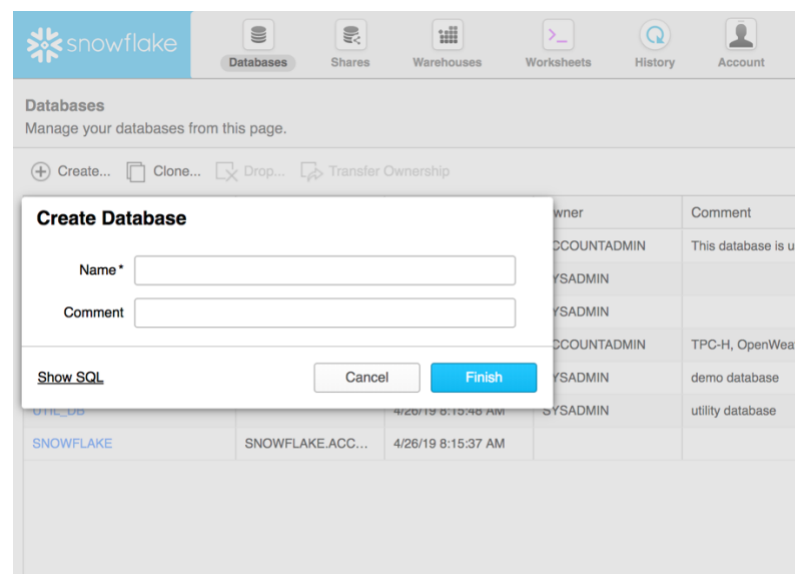
## 6. Database Setup

We would create a new database “CHURN\_DATA\_ANALYSIS” and a new schema “RECENT”.

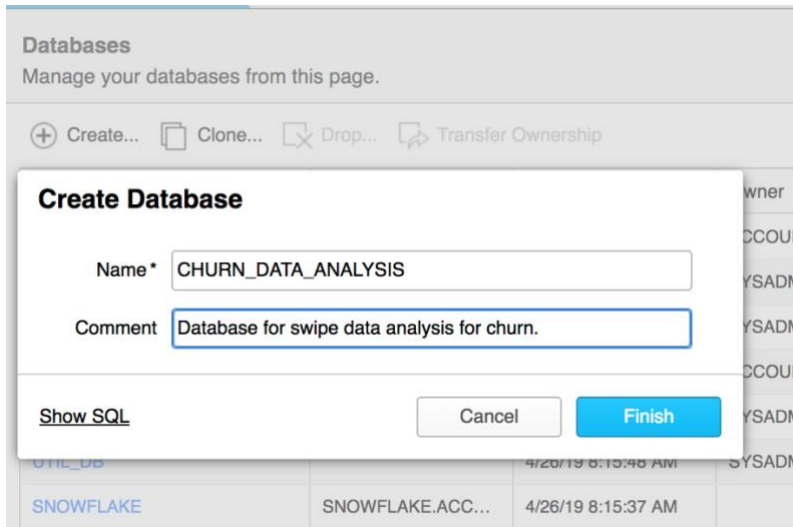
From the Snowflake UI, click on the dropdown arrow on the top right hand side of the screen below your username. Select “Switch Role” and select “ACCOUNTADMIN”.



From the Snowflake UI, click the Database tab at the top, click on “+Create”.



Provide the database “Name\*” as “CHURN\_DATA\_ANALYSIS” and comment as “Database for swipe data analysis for churn”. Then click on Finish.



**Databases**  
Manage your databases from this page.

+ Create... Clone... Drop... Transfer Ownership

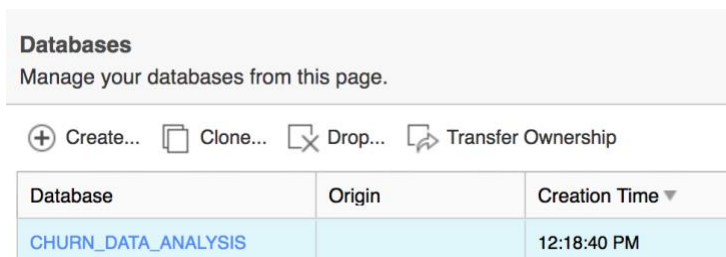
**Create Database**

Name \* CHURN\_DATA\_ANALYSIS

Comment Database for swipe data analysis for churn.

Show SQL Cancel Finish

We will see the database in the table under the Databases tab as below.

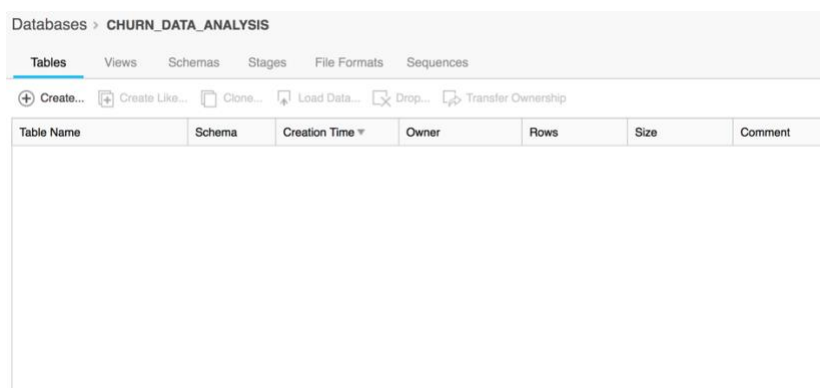


**Databases**  
Manage your databases from this page.

+ Create... Clone... Drop... Transfer Ownership

Database	Origin	Creation Time ▼
CHURN_DATA_ANALYSIS		12:18:40 PM

Click on the CHURN\_DATA\_ANALYSIS Database. We will see the below screen.



Databases > CHURN\_DATA\_ANALYSIS

Tables Views Schemas Stages File Formats Sequences

+ Create... Create Like... Clone... Load Data... Drop... Transfer Ownership

Table Name	Schema	Creation Time ▼	Owner	Rows	Size	Comment
------------	--------	-----------------	-------	------	------	---------

Now click on the Schema Tab.

Databases > CHURN\_DATA\_ANALYSIS

Tables Views **Schemas** Stages File Formats Sequences

+ Create... Clone... Alter... Drop... Transfer Ownership

Schema	Creation Time	Owner	Managed Access	Comment
INFORMATION_SCHEMA	12:21:51 PM			Views describing the contents of schemas in this database
PUBLIC	12:18:40 PM	ACCOUNTADMIN		

Click on “+Create” & type Schema “Name\*” as “Recent” and “Comment” as “Schema to store and analyse recent swipe data for churn.” and click on Finish.

Databases > CHURN\_DATA\_ANALYSIS

Tables Views **Schemas** Stages File Formats Sequences

**Create Schema**

Name \* RECENT

Comment Schema to store and analyse recent swipe data for churn.

☐ Managed Access

[Show SQL](#) Cancel Finish

You will see the Schema name RECENT in the table on the Schema tab of the Main Databases Tab for CHURN\_DATA\_ANALYSIS as below.

Databases > CHURN\_DATA\_ANALYSIS

Tables Views **Schemas** Stages File Formats Sequences

+ Create... Clone... Alter... Drop... Transfer Ownership

Schema	Creation Time	Owner	Managed Access	Comment
INFORMATION_SCHEMA	12:26:51 PM			Views describing
RECENT	12:26:50 PM	ACCOUNTADMIN		Schema to store
PUBLIC	12:18:40 PM	ACCOUNTADMIN		

## 7. Data Preparation for loading into Snowflake

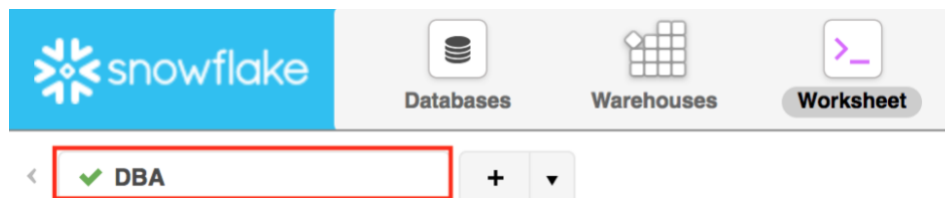
Let's start by loading the structured data into Snowflake. For this exercise, we will be using the COPY command to initiate bulk loading into a table called SWIPE\_DATA.

This section will walk you through the steps to:

- Create a SWIPE\_DATA table in the schema RECENT to store the data.
- Create external stage to specify the AWS S3 location of the CSV files.
- Create a file format for the CSV files.

### Create SWIPE\_DATA table.

From the Snowflake UI, click the Worksheet tab at the top and rename the worksheet "DBA"



First, we need to set the context appropriately within Worksheet. In the top right, click on the drop-down arrow and select the following:

Role: ACCOUNTADMIN  
Database: CHURN\_DATA\_ANALYSIS  
Schema =RECENT.

Copy the text below and paste it into the blank worksheet to create a table:

```
Use ACCOUNTADMIN ;
create table RECENT.SWIPE_DATA(
CustomerKey number(10,0),
Age number(10,0),
Gender varchar(10),
Segment varchar(100),
LocationId number(10,0),
SwipeDate datetime,
SwipeGUID varchar(100),
SwipeHour int,
SwipeMinute int
);
```

Select the command and click Run or hit Ctrl/Cmd+Enter to run the queries.

Run Query  
(Cmd/Ctrl + Return)

▶ Run
☐ All Queries
Saved a few seconds ago

```

1 create table RECENT.SWIPE_DATA(
2 CustomerKey number(10,0),
3 Age number(10,0),
4 Gender varchar(10),
5 Segment varchar(100),
6 LocationId number(10,0),
7 SwipeDate datetime,
8 SwipeGUID varchar(100),
9 SwipeHour int,
10 SwipeMinute int
11 );
12

```

Results
Data Preview

✓ Query ID SQL 410ms 1 rows

Filter result...
Download
Copy

Row	status
1	Table SWIPE_DATA successfully created.

Verify that your table SWIPE\_DATA has been created. At the top, go to the Databases tab and navigate to the CHURN\_DATA\_ANALYSIS database. You should see your newly created SWIPE\_DATA table with your schema assigned to it:

Databases > CHURN\_DATA\_ANALYSIS

Tables Views Schemas Stages File Formats

+ Create... + Create Like... Clone... Load Data...

Table Name	Schema	Creation Time ▼
SWIPE_DATA	RECENT	1:06:03 PM

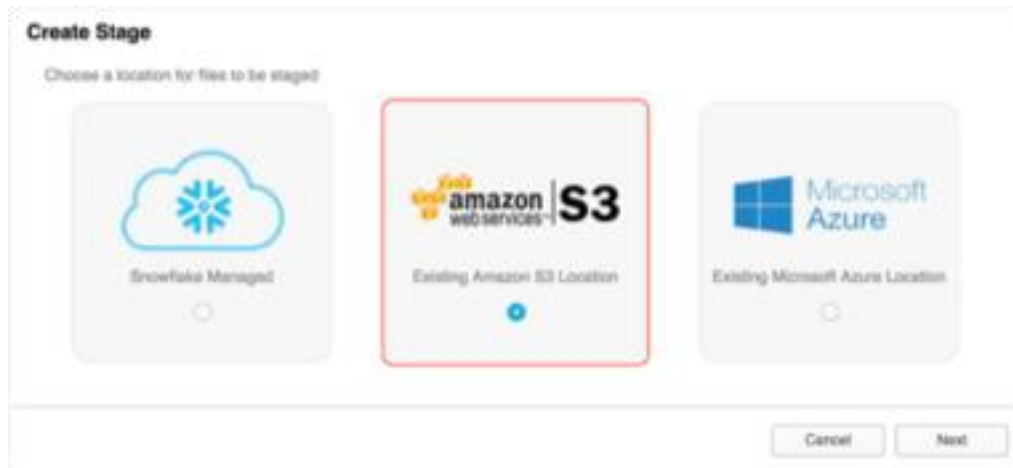
## Create an External Stage

We are working with comma-delimited data that has already been staged in an external S3 bucket. Before we can use this data, we first need to create a Stage that specifies the location of our external bucket and our AWS credentials.



From the Databases tab, under CHURN\_DATA\_ANALYSIS go to Stages and then click “Create...”. Before we can use this data, we first need to create a stage that specifies the location of our external bucket and our AWS credentials

Select Existing Amazon S3 Location:



Use the following settings:

Name	CHURN_DATA_CSV
URL	s3://acro-snowflake-workshop/CSV
AWS Key ID:	
AWS Secret Key:	
Schema	RECENT

Now let’s take a look at the contents of the stage. Switch back to Worksheet and execute the following statement:

```
list @CHURN_DATA_CSV
```

Results

Data Preview

↩ Open History

✓ Query ID

SQL

159ms

18 rows

Filter result...

⬇

Copy

Columns ▾

⌵

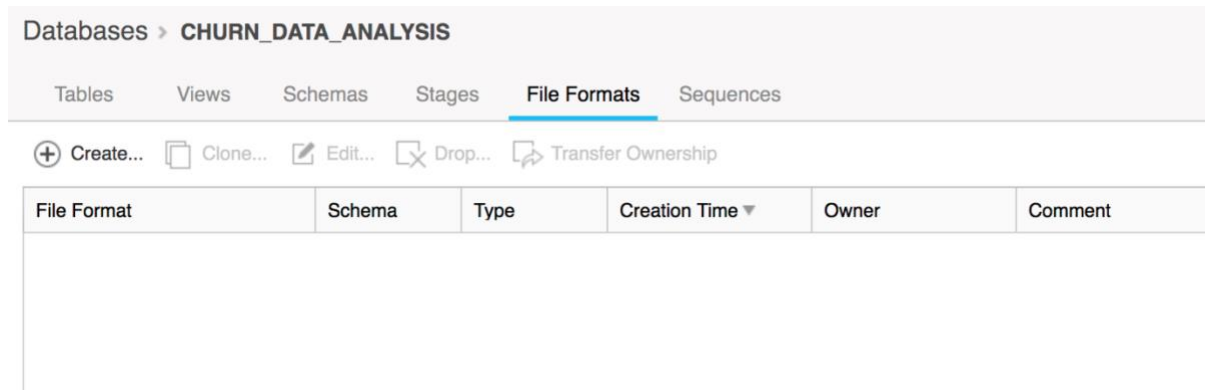
Row	name	size	md5	last_modified
1	s3://dktemp11/dataunload/CSV/data_0_0_0.csv.gz	205712375	3a3b465058fb2b4d18f70a7e5f29d32d...	Thu, 9 May 2019 07:59:05 GMT
2	s3://dktemp11/dataunload/CSV/data_0_1_0.csv.gz	216004224	b4dcaca02459129e5fe39b724a26eb8...	Thu, 9 May 2019 07:59:05 GMT
3	s3://dktemp11/dataunload/CSV/data_0_2_0.csv.gz	204725453	e4fd33ab6715758f2f94279b84c05d46...	Thu, 9 May 2019 07:59:05 GMT

## Create a File Format

Before we can load the data into Snowflake, we have to create a file format that matches its

structure.

From the Databases tab, under CHURN\_DATA\_ANALYSIS, select File Formats



Create a File Format called SWIPE\_CSV with the following properties:

Name: **SWIPE\_CSV**  
Schema Name: **RECENT**  
Format Type: **CSV**  
Compression Method: **Gzip**  
Column separator: **Comma**  
Row separator: **New Line**  
Header lines to skip: **0**  
Field optionally enclosed by: **Double Quote**  
Null string:  
≤Trim space before and after  
≤Error on Column Mismatch

*Leave the rest of the settings as they are*

Details as below:

### Create File Format

Name *	<input type="text" value="SWIPE_CSV"/>
Schema Name	<input type="text" value="RECENT"/>
Format Type	<input type="text" value="CSV"/>
Compression Method	<input type="text" value="Gzip"/>
Column separator	<input type="text" value="Comma"/>
Row separator	<input type="text" value="New Line"/>
Header lines to skip	<input type="text" value="0"/>
Field optionally enclosed by	<input type="text" value="None"/>
Null String	<input type="text"/>
<input checked="" type="checkbox"/> Trim space before and after	

[Show SQL](#)

### Create File Format

Header lines to skip	<input type="text" value="0"/>
Field optionally enclosed by	<input type="text" value="None"/>
Null String	<input type="text"/>
<input checked="" type="checkbox"/> Trim space before and after	
<input checked="" type="checkbox"/> Error on Column Count Mismatch	
Escape Character	<input type="text" value="None"/>
Escape Unenclosed Field	<input type="text" value="None"/>
Date Format	<input type="text" value="Auto"/>
Timestamp Format	<input type="text" value="Auto"/>
Comment	<input type="text"/>

[Show SQL](#)

The file format is created as below:

Databases > CHURN\_DATA\_ANALYSIS

Tables Views Schemas Stages File Formats Sequences

+

 Create...

Clone...

Edit...

Drop...

Transfer Ownership

File Format	Schema	Type	Creation Time ▾	Owner	Comment
SWIPE_CSV	RECENT	CSV	11:42:54 PM	ACCOUNTADMIN	

## 8. Loading data

In this section we would do the actual data of the gym swipe data. The data has been exported and pre-staged for you in the S3 bucket. It consists of information such as: member profile, swipe date, swipe times, swipe location, etc.

Below is the snippet from one of the sample CSV files. It is in a comma-delimited format with double quote enclosing and a single header line.

```
2930220,27,F,Health Neglecters,34,2018-12-01,5bae2bfc-e528-4fd4-99fc-48613d5d7b2f,14,36
2930220,27,F,Health Neglecters,34,2018-12-14,5504ccf8-8660-4af4-b12c-ad7734bc676e,18,45
2644670,49,M,Health Neglecters,20,2018-12-16,4f6fdc9-d0e-4b30-ba34-ec3bbd239e19,10,25
2644670,49,M,Health Neglecters,20,2018-12-19,a943b3e6-480f-4ca5-8e6a-e44b62f4a3ad,17,43
2644670,49,M,Health Neglecters,20,2018-12-20,c90f3f58-f257-480b-b64f-5c3af1b27127,15,38
2644670,49,M,Health Neglecters,20,2018-12-27,9fe8cd36-7ca6-4723-af43-245f407687a0,2,5
2644670,49,M,Health Neglecters,20,2018-12-28,2aec938d-237c-4455-9d49-a5e8f24cdb91,7,17
2644670,49,M,Health Neglecters,20,2018-12-29,24302971-44f2-4c88-9e82-16addb7cb978,16,41
2644670,49,M,Health Neglecters,20,2018-12-02,631bf342-1dd1-49e7-a60a-544d5aa7e27f,19,50
2644670,49,M,Health Neglecters,20,2018-12-03,5519a3c8-1681-48de-9afc-776c15174d04,7,18
2644670,49,M,Health Neglecters,20,2018-12-04,a7aa88b9-fb7d-47d7-ae3e-6747d2b18372,8,20
2644670,49,M,Health Neglecters,20,2018-12-06,937e5406-d91c-4253-9c41-95e8614325aa,19,49
2644670,49,M,Health Neglecters,20,2018-12-07,d6c72338-4e9c-4b5d-ab3c-426227d31f8e,20,51
2644670,49,M,Health Neglecters,20,2018-12-08,12da4977-4866-4d12-8109-6cdf321a65e9,10,26
2644670,49,M,Health Neglecters,20,2018-12-09,f6b9960c-3c20-48bd-b9c0-864f8239dbf5,3,9
2644670,49,M,Health Neglecters,20,2018-12-11,fac3f7d0-8ec2-4c1d-8418-cd9410844e01,9,24
2644670,49,M,Health Neglecters,20,2018-12-12,88b613b7-bee6-4f0b-87d9-27326211d361,3,8
2644670,49,M,Health Neglecters,20,2018-12-13,913e0e0a-66a9-489e-ac53-7ad1de98c7e2,20,50
2644670,49,M,Health Neglecters,20,2018-12-15,0fcbf039-914c-4ecb-ba1d-c8e12b98a587,16,42
2644670,49,M,Health Neglecters,20,2018-12-17,38f286c7-1bbe-455d-bf2a-60fb48c2617e,5,13
2644670,49,M,Health Neglecters,20,2018-12-21,c438ba86-a795-4364-8c2d-2de8fe489130,1,4
2644670,49,M,Health Neglecters,20,2018-12-22,5ca1cc47-608c-48ec-a4d8-1afcbfc7809c,0,1
```

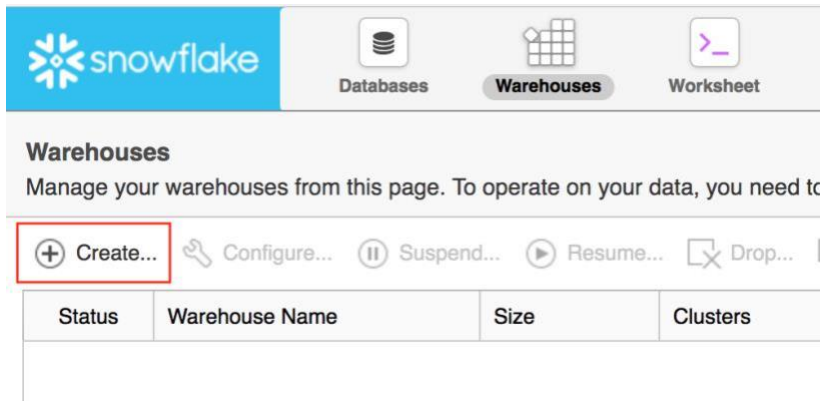
This section will walk you through the steps to:

- Create warehouse of size xsmall.
- Change the context for running queries to use the newly created warehouse.
- Load the csv files in the table SWIPE\_DATA.
- Verify if the data is loaded.

### Create warehouse

For loading data, compute power is needed. Snowflake's compute nodes are called Warehouses and they can be dynamically sized according to the workload.

Navigate to the Warehouses tab and click on Create.



The screenshot shows the Snowflake 'Warehouses' tab in the management console. At the top, there are three tabs: 'Databases', 'Warehouses' (which is selected), and 'Worksheet'. Below the tabs, the 'Warehouses' section is titled 'Warehouses' with a subtitle 'Manage your warehouses from this page. To operate on your data, you need to...'. A row of action buttons is visible: '+ Create...' (highlighted with a red box), 'Configure...', 'Suspend...', 'Resume...', and 'Drop...'. Below these buttons is a table with the following headers: 'Status', 'Warehouse Name', 'Size', and 'Clusters'. The table body is currently empty.

Create a new warehouse called DATA\_LOAD\_WH with the following settings:

### Configure Warehouse

Name DATA\_LOAD\_WH

Size

Learn more about virtual warehouse sizes [here](#)

Maximum Clusters

Multi-cluster warehouses improve the query throughput for high concurrency workloads.

Minimum Clusters

The number of active clusters will vary between the specified minimum and maximum values, based on number of concurrent users/queries.

Scaling Policy

The policy used to automatically start up and shut down clusters.

Auto Suspend

The maximum idle time before the warehouse will be automatically suspended.

☒ Auto Resume [?](#)

Comment

[Show SQL](#)

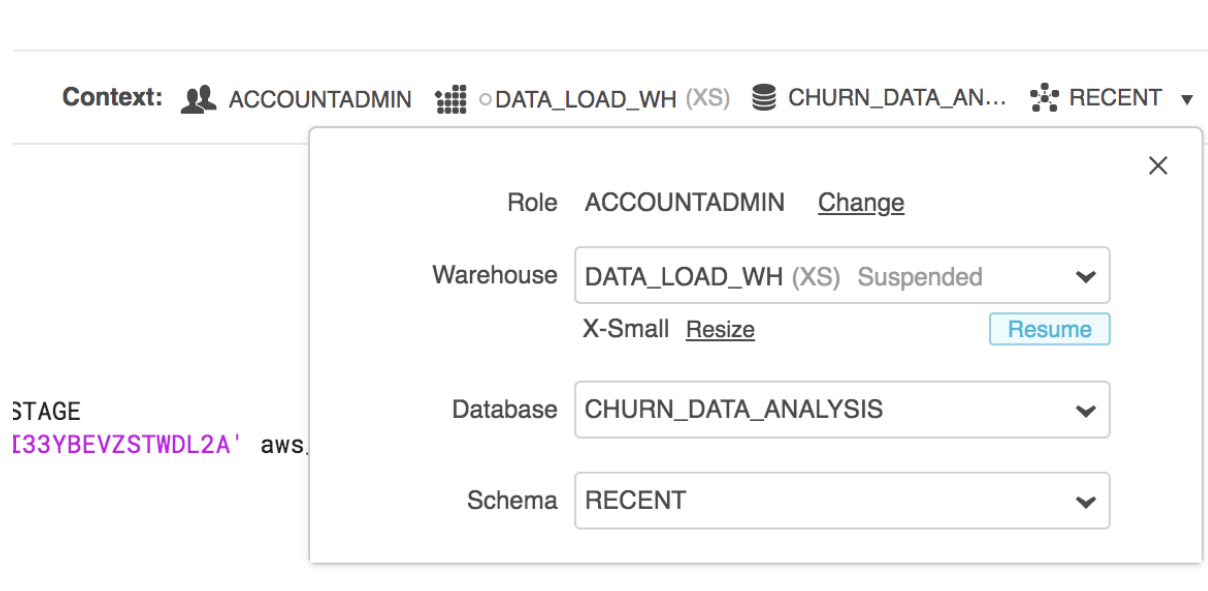
## Change the context

We need to change the context for the worksheet tab to ensure that we use the newly created warehouse for running queries against CHURN\_DATA\_ANALYSIS database->RECENT schema.

Navigate to the Worksheet tab and run the below SQL query.

```
use warehouse DATA_LOAD_WH;  
use database CHURN_DATA_ANALYSIS;  
use schema RECENT;
```

Once the queries have been executed navigate to the top right hand section under your schema name click on the down arrow head.



## Load the data

We would load the CSV files staged in the S3 bucket data using the newly created warehouse in the SWIPE\_DATA table.

Navigate to the Database Tab & select the database CHURN\_DATA\_ANALYSIS.

## Databases

Manage your databases from this page.

 Create... 
  Clone... 
  Drop... 
  Transfer Ownership

Database	Origin	Creation Time ▼	Owner
<a href="#">CHURN_DATA_ANALYSIS</a>		12:18:40 PM	ACCOUNTADMIN

Under the Tables tab for the database, select the table SWIPE\_DATA.

## Databases > CHURN\_DATA\_ANALYSIS

Tables

Views

Schemas

Stages

File Formats

Sequences






 Create... 
  Create Like... 
  Clone... 
  Load Data... 
  Drop... 
  Transfer Ownership

Table Name	Schema	Creation Time ▼	Owner
<a href="#">SWIPE_DATA</a>	RECENT	1:06:03 PM	ACCOUNTADMIN

Once selected, click on the Load Table.

## Databases > CHURN\_DATA\_ANALYSIS > SWIPE\_DATA (RECENT)

Tables


Views

Schemas

Stages

File Formats

Sequences

 Load Table

Column Name	Ordinal ▲	Type	Nullable	Default
CUSTOMERKEY	1	NUMBER(10,0)	true	NULL
AGE	2	NUMBER(10,0)	true	NULL
GENDER	3	VARCHAR(10)	true	NULL
SEGMENT	4	VARCHAR(100)	true	NULL
LOCATIONID	5	NUMBER(10,0)	true	NULL
SWIPEDATE	6	TIMESTAMP_NTZ(9)	true	NULL
SWIPEGUID	7	VARCHAR(100)	true	NULL
SWIPEHOUR	8	NUMBER(38,0)	true	NULL
SWIPEMINUTE	9	NUMBER(38,0)	true	NULL



You will get a new window popped up to configure the details for loading the data.

Configure it as below:

1. Warehouse: DATA\_LOAD\_WH

### Load Data

Warehouse

Source Files

File Format

Load Options

Which warehouse do you want to use to load the files?

DATA\_LOAD\_WH

CancelNext

2. Load files from S3 Bucket: CHURN\_DATA\_CSV —

**S3 Bucket to be used - > s3://acro-snowflake-workshop/CSV**

Note: Example S3 bucket used in below screenshot.

### Load Data

Warehouse

Source Files

File Format

Load Options

From where do you want to load files?

☐ Load files from your computer

Select Files...

☒ Load files from S3 bucket

StageCHURN\_DATA\_CSV -- s3://dktemp11/dataunload/CSV

Path

CancelBackNext

3. File Format: SWIPE\_CSV

## Load Data

Warehouse
Source Files
File Format
Load Options

SWIPE\_CSV

[Show SQL](#)
Cancel
Back
Next
Load

4. Load Options: Continue loading valid data from the file.

## Load Data

Warehouse
Source Files
File Format
Load Options

What should the load do if it encounters an error while parsing a file?

☐ Do not load any data in the file  
☐ Stop loading, rollback and return the error  
☐ Do not load any data in the file if the error count exceeds:  
Threshold

☒ Continue loading valid data from the file

[Show SQL](#)
Cancel
Back
Load

Click on the Option Show SQL and below windows pops up.

## SQL

1
COPY INTO "CHURN\_DATA\_ANALYSIS"."RECENT"."SWIPE\_DATA" FROM  
'@"CHURN\_DATA\_ANALYSIS"."RECENT"."CHURN\_DATA\_CSV"' FILE\_FORMAT =  
'"CHURN\_DATA\_ANALYSIS"."RECENT"."SWIPE\_CSV"' ON\_ERROR = 'CONTINUE'  
PURGE = FALSE;

Select SQL
Close

Copy the SQL on to a note pad and save it.

Now go back to the previous screen by clicking on Close.

Now click on Load and after few seconds we would get below summary of data loads and click on OK.

## Load Results

Loaded	File	Rows Parsed	Rows Loaded
✓	s3://dktemp11/dataunload/CSV/data_0_2...	7798340	7798340
✓	s3://dktemp11/dataunload/CSV/data_0_0...	7836991	7836991
✓	s3://dktemp11/dataunload/CSV/data_0_3...	7807876	7807876
✓	s3://dktemp11/dataunload/CSV/data_0_5...	7969360	7969360
✓	s3://dktemp11/dataunload/CSV/data_0_1...	8217893	8217893

---

OK

## Load Results

Loaded	File	Rows Parsed	Rows Loaded
✓	s3://dktemp11/dataunload/CSV/data_0_5...	7969360	7969360
✓	s3://dktemp11/dataunload/CSV/data_0_1...	8217893	8217893
✓	s3://dktemp11/dataunload/CSV/data_0_4...	8211467	8211467
✓	s3://dktemp11/dataunload/CSV/data_0_6...	8222475	8222475
✓	s3://dktemp11/dataunload/CSV/data_0_7...	8226694	8226694

OK

## Validate the data loaded

Check the count of records loaded:

1. Copy the below SQL query in the worksheet and click on Run (or Ctrl+Enter).

```
select count(1) FROM RECENT.SWIPE_DATA;
```

Below is the query output.

Results

Data Preview

↶ Open History

✓

Query ID

SQL

205ms

1 rows

Filter result...

⬇

Copy

Columns ▾

⌵

Row	COUNT(1)
1	64291096

2. Check for sample records loaded:

```
Select * from RECENT.SWIPE_DATA limit 10;
```

Below is the query output.

ResultsData PreviewOpen History

✓ Query ID SQL 228ms 10 rows

Filter result...DownloadCopy

Columns

Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	LOCATIONID	SWIPEDATE	SWIPEGUID ↓	SWIPEHOUR	SWIPEMINUTE
1	3019737	32	F	Hurried Healthies	47	2018-06-25 00:00:00.000	bcc128b0-2ab1-4a4c-9b9e-df3a739a4694	7	20
4	3019737	32	F	Hurried Healthies	47	2018-06-25 00:00:00.000	71cbfbc5-3619-47f2-b80b-664d7ae15111	4	11
3	3019737	32	F	Hurried Healthies	47	2018-06-20 00:00:00.000	6794fe8b-a73a-4c2c-a9cf-89c98c1318cc	10	27
6	3019737	32	F	Hurried Healthies	47	2018-06-29 00:00:00.000	47cd73ce-5ab7-403b-b8dd-a66e44373a1	19	48
8	3019737	32	F	Hurried Healthies	47	2018-07-08 00:00:00.000	3fe22055-8062-4430-a46a-6c89fe6961ea	9	24
10	3019737	32	F	Hurried Healthies	47	2018-07-14 00:00:00.000	3172cc30-fb34-4d21-8ad1-1568f8fc4b36	15	39
2	3019737	32	F	Hurried Healthies	47	2018-06-18 00:00:00.000	241da35d-e4c3-4171-8af8-0fd1dd499bf8	23	59
5	3019737	32	F	Hurried Healthies	47	2018-06-27 00:00:00.000	0ffe45be-f373-472e-b694-8a0696dd1eb9	9	22
9	3019737	32	F	Hurried Healthies	47	2018-07-12 00:00:00.000	0740cbf6-3617-495b-8685-6886334820b3	8	21
7	3019737	32	F	Hurried Healthies	47	2018-07-03 00:00:00.000	02d7f1155-b4ab-4967-943e-cc17d05ee3fd	16	41

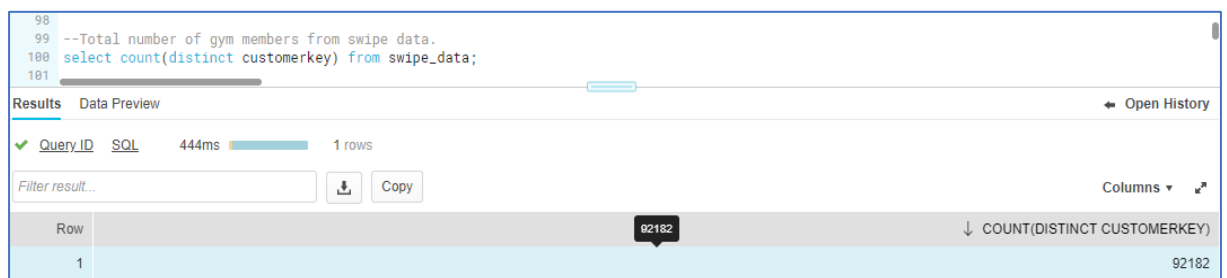
## 9. Running Analytical Queries:

Till now we have loaded the CSV data in the Snowflake table. Now let's run some queries against the table.

### Get basic counts of gym members from swipe data:

```
select count(distinct customerkey) from swipe_data;
```

Output:



The screenshot shows a Snowflake query interface. The query is: `--Total number of gym members from swipe data. select count(distinct customerkey) from swipe_data;`. The results tab is active, showing a single row with the value 92182. The table has two columns: 'Query ID' and 'SQL'. The 'Query ID' column shows '444ms' and '1 rows'. The 'SQL' column shows the query text. The results table has a header row with 'Row' and 'COUNT(DISTINCT CUSTOMERKEY)'. The first row shows '1' and '92182'.

Row	COUNT(DISTINCT CUSTOMERKEY)
1	92182

### Most busiest day of the week

Let's get the busiest day of the week.

Copy the below query on the worksheet tab and click on Run (or hit Ctrl & Enter)

```
select  
dayname(swipedate) as "day of week", count(distinct customerkey) as "num of  
customer"  
from swipe_data  
group by 1 order by 2 desc;
```

Output

```

182 |
183 | --Most busiest day of the week
184 | select
185 | dayname(swipedate) as "day of week", count(distinct customerkey) as "num of customer"
186 | from swipe_data
187 | group by 1 order by 2 desc;
188 |

```

Row	day of week	num of customer
1	Sat	92182
2	Tue	91754
3	Thu	91754
4	Wed	91754
5	Mon	91754
6	Sun	91754
7	Fri	91732

## Active gym members

Copy the below query on the worksheet tab and click on Run (or hit Ctrl & Enter)

*select count(distinct swipeguid),customerkey from swipe\_data group by customerkey order by count(distinct swipedate) desc ;*

*Note: Time taken 13 - 16.23 seconds.*

```

46 | select COUNT(DISTINCT SWIPEGUID),CUSTOMERKEY
47 | FROM SWIPE_DATA GROUP BY CUSTOMERKEY
48 | order by count(distinct swipedate) desc;
49 |
50 |
51 |
52 |
53 |
54 |

```

Row	COUNT(DISTINCT SWIPEGUID)	CUSTOMERKEY
1	757	3019312
2	757	1387915
3	759	1005694

**Alter Data Warehouse Size** using below command:

*ALTER WAREHOUSE DATA\_LOAD\_WH SET WAREHOUSE\_SIZE = 'XLARGE';*

*Run below once again and observe the time take now:*

*select count(distinct swipeguid),customerkey from swipe\_data group by customerkey order by count(distinct swipedate) desc ;*



The screenshot shows a Snowflake query interface. At the top, a SQL query is displayed: `select COUNT(DISTINCT SWIPEGUID), CUSTOMERKEY FROM SWIPE_DATA GROUP BY CUSTOMERKEY order by count(distinct swipedate) desc;`. Below the query, a performance summary box shows: Total Duration 2.79s, Compilation 21ms, Execution 2.77s, Compiling SQL 21ms, Executing (serial) 6ms, Wait process group 13ms, and Executing (warehouse) 2.33s. The 'Results' tab is active, showing a table with 92,183 rows. The table has two columns: 'COUNT(DISTINCT SWIPEGUID)' and 'CUSTOMERKEY'. The first four rows are visible:

Row	COUNT(DISTINCT SWIPEGUID)	CUSTOMERKEY
1	757	3019312
2	757	1387915
3	759	1005694
4	749	2496247

## Top 10 active gym members

Copy the below query on the worksheet tab and click on Run (or hit Ctrl & Enter)

*select customerkey, count(distinct swipedate) from swipe\_data group by customerkey order by count(distinct swipedate) desc limit 10 ;*

Row	CUSTOMERKEY	COUNT(DISTINCT SWIPEDATE)
1	3019312	701
2	1387915	696
3	1005694	694
4	2683953	692
5	2496247	692
6	2093849	692
7	2990543	691
8	2820493	690
9	2864023	690
10	2223067	690



## 10. Data Load and Errors

We will now try to load the file LOCATION.csv.gz into a new table called LOCATION.

Below are the details of the table and file.

```
CREATE TABLE RECENT.LOCATION
  (LOCATION_KEY          VARCHAR(50)          ,
   POSTCODE            VARCHAR(50)          ,
   LATITUDE            VARCHAR(50)          ,
   LONGITUDE           VARCHAR(50)          ,
   ZONE                VARCHAR(50)          ,
   COUNTRY             VARCHAR(50)          ,
   LOCATION_NAME       VARCHAR(50)
);
```

File is location at the S3 bucket 's3://acro-snowflake-workshop' by name of LOCATION.csv.gz  
Sample records from the file with header:

LOCATION_KEY	POSTCODE	LATITUDE	LONGITUDE	ZONE	COUNTRY	LOCATION_NAME
1,0X1 1AN	51.749308	-1.259773	0xford	UK	Carfax	
2,0X9 7JA	51.706866	-1.003401	0xford	England	Haseley Brook	
3,BA12 0PT	51.146189	-1.993137	Greater London	England	Salisbury	
4,CB1 0SA	52.204838	0.119253	Cambridge	England	Market	
5,CB1 0AD	52.192267	0.137208	Cambridge	England	Coleridge	

### Create file stage, file format and table for data load

Copy below SQL queries and click on Run (or hit CTRL & Enter)

--Create File Stage:

```
CREATE OR REPLACE STAGE
"CHURN_DATA_ANALYSIS"."RECENT".CHURN_LOC_DATA_STAGE
URL = 's3://acro-snowflake-workshop'
credentials = (aws_key_id='AKIAII33YBEVZSTWDL2A'
aws_secret_key='UDe0GGWdCC3rmhx2lVG2knMsQeqHNUw9F+YDj+DB');
LIST @CHURN_LOC_DATA_STAGE;
```

--File format for LOCATION

```
CREATE OR REPLACE FILE FORMAT "CHURN_DATA_ANALYSIS"."RECENT".location_csv TYPE =
'CSV'
COMPRESSION = 'gzip' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1
FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE'
TRIM_SPACE = FALSE ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE'
ESCAPE_UNENCLOSED_FIELD = '\134'
```

`DATE_FORMAT = 'AUTO' TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');`

## Load data and identify error

Load the data in the table by running below SQL query.

```
copy INTO recent.location FROM @CHURN_LOC_DATA_STAGE/LOCATION.csv.gz
file_format=location_csv;
```

### Output:

Results Data Preview										
Query ID SQL 1.16s 1 rows										
Filter result...										
Columns										
Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column...
1	s3://dktemp11/data...	PARTIALLY_LOAD...	103	101	103	2	User character len...	7	46	"LOCATION_KEY"...

## Error analysis and reload of data

As seen in the output of the copy command we have got some errors while loading the data. Let us now load the error details into a new table and analyse them.

1. Click on the Open history Hyperlink and then click on the column, Query ID-related to the copy command which we executed.

Results Data Preview										
Query ID SQL 620ms 1 rows										
Filter result...										
Copy Columns										
Row	file	status	rows_parsed	rows_loaded						
1	s3://dktemp1...	PARTIALLY_...	103	102						

History 45										
Status	Duration	Start	End	Query ID	Rows	Bytes Scanned	Cluster	SQL		
✓	620ms	16:09:25	16:09:25	018c230...	102	3.0KB	1	copy_into RECENT.LOCATION_KEY		
✓	278ms	16:09:19	16:09:19	018c230...				truncate_table RECENT.LOCATIO		
✓	387ms	16:08:47	16:08:47	018c230...			1	copy_into CHURN_DATA_ANALYSIS		
✓	289ms	16:08:35	16:08:35	018c230...				truncate_table RECENT.LOCATIO		
✓	688ms	16:03:36	16:03:37	018c230...	102	3.0KB	1	copy_into RECENT.LOCATION_KEY		
✗	13ms	16:03:30	16:03:30	018c230...				CREATE TABLE RECENT.LOCATION_		
✓	110ms	16:03:30	16:03:30	018c230...				CREATE OR REPLACE STAGE "CHUR		

You will see the below window: Copy the Query ID.

History 4:09:25 PM for 620ms										
Last refreshed 4:09:49 PM Auto refresh										
Details Profile										
Status	Success									
User	CRADLEOFDATA									
Warehouse	DATA_LOAD_WH									
Start Time	4:09:25 PM									
End Time	4:09:25 PM									
Total Duration	620ms									
Scanned Bytes	0									
Rows	102									
Query ID	018c230d-009d-5b22-0000-118900016436									
Session ID	529133577									

SQL Text										
1 copy_into RECENT.LOCATION_KEY from @CHURN_LOC_DATA_STAGE/LOCATION.csv.gz file_format=location_csv on_error='CONTINUE';										

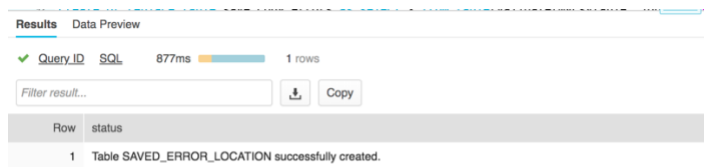
  

Query Result										
Results										
row#	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_li...	first_errc	
1	s3://dktemp1...	PARTIALLY_...	103	102	103	1	User charact...	7	46	

2. Copy the below SQL on the Worksheet tab and replace the placeholder (QUERY\_ID) with the above Query ID. Click on Run (or CTRL & Enter)

create table RECENT.saved\_error\_location as  
select \* from table(validate(LOCATION, job\_id=>'(QUERY\_ID)'));

You will see the below output.

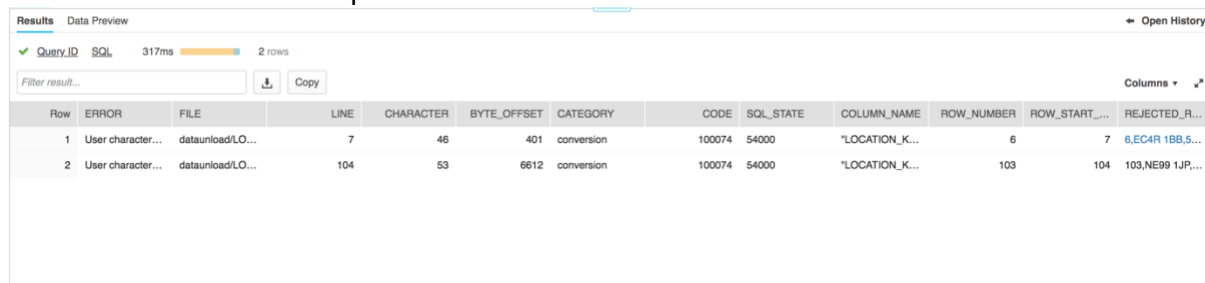


Row	status
1	Table SAVED_ERROR_LOCATION successfully created.

3.Copy the below SQL on the Worksheet tab and replace the placeholder (QUERY\_ID) with the above Query ID. Click on Run (or CTRL & Enter)

select \* from saved\_error\_location;

You will see the below output.



Row	ERROR	FILE	LINE	CHARACTER	BYTE_OFFSET	CATEGORY	CODE	SQL_STATE	COLUMN_NAME	ROW_NUMBER	ROW_START_...	REJECTED_R...
1	User character...	dataunload/LO...	7	46	401	conversion	100074	54000	"LOCATION_K...	6	7	6,EC4R 1BB,5...
2	User character...	dataunload/LO...	104	53	6612	conversion	100074	54000	"LOCATION_K...	103	104	103,NE99 1JP...

4. On analysis we identify that the column Country in the file has value greater than 50 for row 7 & 104, which is the maximum size of Column Country in the table.

Let's say we have fixed this issue by replacing this value with 'UK' and uploaded the corrected file on the S3 bucket with the key name as 'LOCATION\_Corrected.csv.gz'

5. Now let's try to reload the data from the new file

Copy below SQL queries in the Worksheet tab and hit Run or (CTRL & ENTER key).

--Upon Analysis and resolution alongside presenter, run below to re-load/test Location data:  
copy into RECENT.LOCATION from @CHURN\_LOC\_DATA\_STAGE/LOCATION\_Corrected.csv.gz  
file\_format=location\_csv on\_error='CONTINUE';

--Quick Test:

select \* FROM RECENT.LOCATION;

168  
161  
162  
163  
164  
165  
166

```
--Upon Analysis and resolution alongside presenter, run below to re-load/test Location data:  
copy into RECENT.LOCATION from @CHURN_LOC_DATA_STAGE/LOCATION_Corrected.csv.gz file_format=location_csv on_error='CONTINUE';  
  
--Quick Test:  
select * FROM RECENT.LOCATION;
```

ResultsData Preview

↔ Open History

✓ Query ID SQL 249ms 103 rows

Filter result...

Columns

Row: 1	lw	LOCATION_KEY	POSTCODE	LATITUDE	LONGITUDE	ZONE	COUNTRY	LOCATION_NAME
1	1		OX1 1AN	51.749308	-1.259773	Oxford	UK	Carfax
2	2		OX9 7JA	51.706866	-1.003401	Oxford	England	Haseley Brook
3	3		BA12 0PT	51.146189	-1.993137	Greater London	England	Salisbury
4	4		CB1 0SA	52.204838	0.119253	Cambridge	England	Market
5	5		CB1 0AD	52.192267	0.137208	Cambridge	England	Coleridge
6	6		EC4R 1BB	51.511948	-0.09308	Greater London	UK	Vintry

## 11. Working with semi-structured files

Snowflake offers native support for semi-structured data by providing flexible-schema data types for loading JSON, XML, Avro, Parquet, and ORC data without transformation.

This section will walk you through the steps to:

- Load the semi-structured data in the existing table SWIPE\_DATA.
- Query the semi-structured data using SQL dot notation.

### JSON Sample data

Below is the snippet of the files we will load in the SWIPE\_DATA table.

```
{
  "_CustomerKey": "2431149",
  "_CustomerProfile": {
    "_CustomerAge": 28,
    "_CustomerGender": "F",
    "_CustomerSegment": "Happy Families"
  },
  "_LocationId": "17",
  "_SwipeDate": "2018-12-19",
  "_SwipeGuid": "8495a03c-136f-46cc-87d4-1ae9d5be6957",
  "_SwipeHour": 23,
  "_SwipeMinute": 59
}
{
  "_CustomerKey": "2431149",
  "_CustomerProfile": {
    "_CustomerAge": 28,
    "_CustomerGender": "F",
    "_CustomerSegment": "Happy Families"
  },
  "_LocationId": "17",
  "_SwipeDate": "2018-12-20",
  "_SwipeGuid": "4603362c-d537-4dd5-b71b-ce2b0effb626",
  "_SwipeHour": 11,
  "_SwipeMinute": 29
}
```

If we would see that the data is same as what we had in the CSV file but the structure is different.

Let's start creating the file stage, file format for these JSON files.

They are located at the location `s3://acro-snowflake-workshop/JSON` on AWS S3.

## Create JSON file format

Copy below SQL on the worksheet tab and hit Run (or hit CTRL & ENTER)

```
create or replace file format RECENT.CHURN_GZIP_JSON_FORMAT
  type = JSON compression = gzip;
```

We get the below output.

Results		Data Preview	
✓	Query ID	SQL	93ms
Filter result...		Download	Copy
Row	status		
1	File format CHURN_GZIP_JSON_FORMAT successfully created.		

## Create JSON File Stage

Copy below SQL on the worksheet tab and hit Run (or hit CTRL & ENTER)

```
CREATE OR REPLACE STAGE "CHURN_DATA_ANALYSIS"."RECENT".CHURN_DATA_JSON URL
= 's3://acro-snowflake-workshop/JSON';
```

Output

Results		Data Preview	
✓	Query ID	SQL	137ms
Filter result...		Download	Copy
Row	status		
1	Stage area CHURN_DATA_JSON successfully created.		

## Check the files present on the JSON Stage

Copy below sql on the worksheet tab and hit Run (or hit CTRL & ENTER)

```
list @CHURN_DATA_JSON;
```

Row	name	size	md5	last_modified
1	s3://dktemp11/dataunload/JSON/data_0_0.json.gz	240251019	cfc0ae2063ccb129413c848d425d115-15	Thu, 9 May 2019 08:12:30 GMT
2	s3://dktemp11/dataunload/JSON/data_0_1.json.gz	240089017	70eac2a234bb7b609624c26949a5f66c-15	Thu, 9 May 2019 08:12:30 GMT
3	s3://dktemp11/dataunload/JSON/data_0_2.json.gz	252274667	2e5b9e4c3e5c3366620e3ef9a0f12f1-16	Thu, 9 May 2019 08:12:30 GMT
4	s3://dktemp11/dataunload/JSON/data_0_3.json.gz	244094476	a0bed7499d6ba7881a40b1da80f16a2b-15	Thu, 9 May 2019 08:12:30 GMT
5	s3://dktemp11/dataunload/JSON/data_0_4.json.gz	252261817	5a0562c9694eb9a448703cbeb39f0700-16	Thu, 9 May 2019 08:12:30 GMT
6	s3://dktemp11/dataunload/JSON/data_0_5.json.gz	252417325	37ca01b1df899a67e45218142889c4b3-16	Thu, 9 May 2019 08:12:30 GMT
7	s3://dktemp11/dataunload/JSON/data_0_6.json.gz	240673135	8ac2df13234777c154afadc91d680f2c-15	Thu, 9 May 2019 08:12:30 GMT
8	s3://dktemp11/dataunload/JSON/data_0_7.json.gz	252202006	6052f35427ad43b3957df8270c12887-16	Thu, 9 May 2019 08:12:30 GMT

## Create a staging table with VARIANT datatype

Here we would create a table with datatype variant (a universal data type which can hold any type of data including semistructured JSON,XML,Arrays).

Copy below SQL on the worksheet tab and hit Run (or hit CTRL & ENTER)

```
CREATE OR REPLACE TABLE churnjsonstable
(
    json VARIANT
);
```

Output:

**Results** Data Preview

✓ Query ID SQL 73ms  1 rows

Filter result...



Copy

Row	status
1	Table CHURNJSONTABLE successfully created.

## Load the JSON files in the new table created

Copy below SQL on the worksheet tab and hit Run (or hit CTRL & ENTER)

```
COPY into ChurnJsonTable from @CHURN_DATA_JSON
file_format='CHURN_GZIP_JSON_FORMAT';
```

Output:

Results Data Preview Open History

Query ID SQL 2m53s 8 rows

Filter result... Download Copy Columns

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column...
1	s3://dktemp11/data...	LOADED	7806977	7806977	1	0	NULL	NULL	NULL	NULL
2	s3://dktemp11/data...	LOADED	7822131	7822131	1	0	NULL	NULL	NULL	NULL
3	s3://dktemp11/data...	LOADED	7840204	7840204	1	0	NULL	NULL	NULL	NULL
4	s3://dktemp11/data...	LOADED	7951044	7951044	1	0	NULL	NULL	NULL	NULL
5	s3://dktemp11/data...	LOADED	8214297	8214297	1	0	NULL	NULL	NULL	NULL
6	s3://dktemp11/data...	LOADED	8214863	8214863	1	0	NULL	NULL	NULL	NULL
7	s3://dktemp11/data...	LOADED	8213085	8213085	1	0	NULL	NULL	NULL	NULL
8	s3://dktemp11/data...	LOADED	8228495	8228495	1	0	NULL	NULL	NULL	NULL

## Verify the data load

Run below query to get the count of records loaded.

```
select COUNT(*) FROM ChurnJsonTable;
```

Output:

Results Data Preview Open History

Query ID SQL 75ms 1 rows

Filter result... Download Copy Columns

Row	COUNT(1)
1	7813481

```
select * FROM ChurnJsonTable LIMIT 10;
```

Row	JSON
1	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
2	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
3	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
4	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
5	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
6	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
7	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
8	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
9	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...
10	{ "_CustomerKey": 2.874874000000000e+06, "_CustomerProfile": { "_CustomerAge": 2.600000000000000e+01, "_CustomerGender": "M", "_CustomerSegment": "Healthy & Wealthy" }, "_LocationId": 7.300000000000000e+01, "_SwipeDate": "20...

Sample row



## Details

```

1  {
2    "_CustomerKey": 2.8746740000000000e+06,
3    "_CustomerProfile": {
4      "_CustomerAge": 2.6000000000000000e+01,|
5      "_CustomerGender": "M",
6      "_CustomerSegment": "Healthy & Wealthy"
7    },
8    "_LocationId": 7.3000000000000000e+01,
9    "_SwipeDate": "2019-03-16",
10   "_SwipeGuid": "1eb21b0a-a69f-476f-bb0b-6faca506a48b",
11   "_SwipeHour": 7.0000000000000000e+00,
12   "_SwipeMinute": 1.8000000000000000e+01
13 }

```

Copy

Done

## Read sample json data in structured form

Let's now run some queries to parse the JSON table into a structured form.

Copy below sql on the worksheet tab and hit Run (or hit CTRL & ENTER)

```

select
  json:_CustomerKey::float CustomerKey,
  json:_CustomerProfile._CustomerAge::float Age,
  json:_CustomerProfile._CustomerGender::string Gender,
  json:_CustomerProfile._CustomerSegment::string Segment,
  json:_LocationId::float LocationId,
  json:_SwipeGuid::string SwipeGuid,
  timestamp_from_parts(json:_SwipeDate::date ,
    time_from_parts(json:_SwipeHour::float ,
      json:_SwipeMinute::float,0 )) SwipeDate
  from CHURNJSONTABLE
limit 10;

```

Note: We use SQL dot notation to access nested JSON values. It helps to access the values at lower hierarchy of the JSON structure.

Eg. To access the CustomerAge we use

`_CustomerProfile(1st level)._CustomerAge(2nd level)`

Output:

Row	CUSTOMER...	↓ AGE	GENDER	SEGMENT	LOCATIONID	SWIPEGUID	SWIPEDATE
1	1445135	37	M	Hurried Healthies	30	e47fdd5c-1ed6-40e3-afca-5b5fc300f1b2	2019-02-25 04:11:00.000
2	1445135	37	M	Hurried Healthies	30	92b8df3d-701f-4f10-aae0-23c93da9563c	2019-02-27 01:03:00.000
3	1445135	37	M	Hurried Healthies	30	61cd1322456a3-6483-4af4-9ff8-e1f8c2fb8271	2019-02-28 18:47:00.000
4	1445135	37	M	Hurried Healthies	30	322456a3-6483-4af4-9ff8-e1f8c2fb8271	2019-03-01 11:30:00.000
5	1445135	37	M	Hurried Healthies	30	124c318b-f368-4d85-84e6-ff239309837f	2019-03-02 06:15:00.000
6	1445135	37	M	Hurried Healthies	30	775be6c4-8fb8-48e9-a7a0-9f9de26c19fc	2019-03-05 08:22:00.000
7	1445135	37	M	Hurried Healthies	30	e6a3a78a-fecf-4088-b7a5-18d158db9383	2019-03-06 22:57:00.000
8	1445135	37	M	Hurried Healthies	30	a4d4c655-6b44-4957-b28a-81aa8de5b652	2019-03-08 05:14:00.000
9	1445135	37	M	Hurried Healthies	30	3c25b920-bf1c-41e6-bbfc-bdba89a06bef	2019-03-16 19:50:00.000
10	1445135	37	M	Hurried Healthies	30	9ae7bf55-535a-45df-99a7-90612d7438d9	2019-03-19 03:10:00.000

## Create view to query the JSON data into structured form

Using the above query, we can create a view to query JSON into structured form.

Query: Copy paste on the worksheet tab and click on Run (or CTRL & ENTER key)

```
CREATE OR replace VIEW v_churn_data AS
(
select
  json:_CustomerKey::float CustomerKey,
  json:_CustomerProfile._CustomerAge::float Age,
  json:_CustomerProfile._CustomerGender::string Gender,
  json:_CustomerProfile._CustomerSegment::string Segment,
  json:_LocationId::float LocationId,
  json:_SwipeGuid::string SwipeGuid,
  timestamp_from_parts(json:_SwipeDate::date ,
    time_from_parts(json:_SwipeHour::float ,
      json:_SwipeMinute::float,0 )) SwipeDate
  from CHURNJSONTABLE
);
```

Output:

Results		Data Preview	
✓	Query ID	SQL	242ms 1 rows
Filter result...		Download	Copy
Row	status		
1	View V_CHURN_DATA successfully created.		

## Get location details for swipe data

Let's say we have to get the latitude and longitude for this data. To achieve the output we need to join the `v_CHURN_DATA` (view on semi-structured data) with `LOCATION` table which we loaded earlier.

Query: (Copy paste on the worksheet tab and click on Run (or CTRL & ENTER key)

```
Select * from v_CHURN_DATA v join LOCATION L on v.LOCATIONID = L.LOCATION_KEY limit 10;
```

Output :

Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	LOCATIONID	SWIPEGUID	SWIPEDATE	LATITUDE	LONGITUDE
1	2692097	31	F	Fit & Focused	73	48ef5d17-823e-4214-...	2019-02-04 06:16:00...	52.517877	-1.891111
2	2692097	31	F	Fit & Focused	73	5ced5558-ed51-4309-...	2019-02-05 22:57:00...	52.517877	-1.891111
3	2692097	31	F	Fit & Focused	73	1c5edddb-66ad-4eeb-...	2019-02-17 01:02:00...	52.517877	-1.891111
4	2692097	31	F	Fit & Focused	73	19cba097-9f67-423d-...	2019-02-18 02:07:00...	52.517877	-1.891111
5	2692097	31	F	Fit & Focused	73	f78f181-5c44-4eaa-a...	2019-02-20 14:37:00...	52.517877	-1.891111
6	2692097	31	F	Fit & Focused	73	0f436a44-cba2-4754-...	2019-02-26 03:08:00...	52.517877	-1.891111
7	2692097	31	F	Fit & Focused	73	acc3de17-bddc-44da-...	2019-03-03 18:47:00...	52.517877	-1.891111
8	2692097	31	F	Fit & Focused	73	7d024a1d-cb95-4151-...	2019-03-04 14:36:00...	52.517877	-1.891111
9	2692097	31	F	Fit & Focused	73	6f2973ed-2598-403d-...	2019-03-13 12:32:00...	52.517877	-1.891111
10	2692097	31	F	Fit & Focused	73	3223051f-25eb-43cc-...	2019-03-18 03:09:00...	52.517877	-1.891111

## Insert the data into the existing SWIPE\_DATA table

Query: (Copy paste on the worksheet tab and click on Run(or CTRL & ENTER key)

```
INSERT INTO recent.swipe_data
(customerkey, age, gender, segment, locationid, swipeguid, swipedate)
SELECT customerkey, age, gender, segment, locationid, swipeguid, swipedate FROM
v_churn_data;
```

Output:

Row	number of rows inserted
1	7613481

## 12. DML Operations on data

Now we need to prepare and share the swipe data with the ML team for churn analysis.

In this section,

- We would insert, update data in a new table.
- Drop/ undrop table.
- Time travel.

### Create and Insert data in a new table (using create table as an option)

We analyzed and identified the following columns from swipe\_data to be shared with the ML team.

CustomerKey,  
Age,  
Gender,  
Segment,  
TotalVisits(Calculated as the count of distinct swipedate)

So based on above requirements we will create a table and insert records into it.

Query: (Copy paste on the worksheet tab and click on Run(or CTRL & ENTER key)

```
Create OR REPLACE table CHURN_DATA as
select
S.CustomerKey,S.Age,S.Gender,S.Segment,L.POSTCODE,L.LATITUDE,L.LONGITUDE,L.ZONE,L.C
OUNTRY,L.LOCATION_NAME,
COUNT(DISTINCT S.SWIPEDATE) TOTALVISITS
FROM RECENT.SWIPE_DATA S,
RECENT.LOCATION L WHERE S.LOCATIONID = L.LOCATION_KEY
GROUP BY
S.CustomerKey,S.Age,S.Gender,S.Segment,L.POSTCODE,L.LATITUDE,L.LONGITUDE,L.ZONE,L.C
OUNTRY,L.LOCATION_NAME;
```

Output:

Results   Data Preview

✓ Query ID   SQL   10.32s      1 rows



Copy

Row	status
1	Table CHURN_DATA successfully created.

## 2. Check the data loaded

Query: (Copy paste on the worksheet tab and click on Run (or CTRL & ENTER key)

```
select * FROM CHURN_DATA limit 10 ;
```

### Output:

Results Data Preview Open History

✓ Query ID SQL 587ms 10 rows

Filter result... Download Copy Columns ▾

Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	TOTALVISITS	LOCATION_NAME	POSTCODE	COUNTRY
1	2985458	27	M	Fit & Focused	732	Armley	LS13 2PZ	England
2	2387364	45	F	Fit & Focused	756	Bramley & Stanningley	LS28 6HZ	England
3	2387364	39	M	Fit & Focused	699	Mortlake and Barnes Co...	SW15 6LF	England
4	2445702	41	M	Fit & Focused	737	Pentwyn	CF23 9UL	Wales
5	926198	18	F	Happy Families	739	Bramley & Stanningley	LS13 2SU	England
6	2408910	55	M	Fit & Focused	727	Carfax	OX1 1AN	UK
7	2966749	4	M	Fit & Focused	730	Kirkstall	LS5 3QE	England
8	2941788	29	M	Comfy & Complacent	737	Picton	L7 6NG	England
9	753367	58	F	Fit & Focused	717	Castle	SA1 1AF	Wales
10	2521270	30	M	Fit & Focused	711	St James's	WC2R 3LA	England

## 13. Mistakes and how to recover data using undrop, time-travel and zero-cloning

Snowflake provides additional features to recover data should the case of a table getting dropped/alterd or data getting changed occur.



### Drop and undrop table

Let's say someone in the team drops the table. We can recover it by using the undrop command.

Query :

```
--Drop the table
drop table CHURN_DATA;
```

#### Output

Results		Data Preview		
✓	Query ID	SQL	73ms	1 rows
<input type="text" value="Filter result..."/>  				
Row	status			
1	CHURN_DATA successfully dropped.			

--Confirm if the table is dropped

```
Select count(1) from CHURN_DATA;
```

Output:

Results		Data Preview		
✗	Query ID	SQL	29ms	
SQL compilation error: Object 'CHURN_DATA' does not exist.				

```
--Undrop the table
undrop table churn_data;
```

Output:

**Results** Data Preview

✓ Query\_ID SQL 65ms  1 rows



Copy

Row	status
1	Table CHURN_DATA successfully restored.

--Confirm if the table is available back.

*Select count(1) from CHURN\_DATA;*

Output:

**Results** Data Preview

✓ Query\_ID SQL 198ms  1 rows



Copy

↓ Row	COUNT(1)
1	92182

## Time-travel

Let's say someone in the team updates the churn\_data's total visits=0. Then we can use time-travel to recover the data. Time travel can recover data based on the timestamp or Query ID.

Query:

--Wrongly updated a column values

*update churn\_data set totalvisits=0;*

Output:



## Results Data Preview

✓ Query ID SQL 957ms  1 rows




Row	↓ number of rows updated	number of multi-joined rows updated
1	92182	0

Open history tab and open the Query ID that ran the wrong updates.

Eg.

History 50

Columns ▾ ×

Status	Duration	Start	End	Query ID	Rows	Bytes Scanned	Cluster	SQL
✓	957ms	20:15:17	20:15:18	018c2403-00e1-ad9c-0...	92...	762.2KB	1	update churn_data set totalvisits=0;

History > 8:15:17 PM for 957ms

Details

Profile

Status

Success

User

CRADLEOFDATA

Warehouse

DATA\_LOAD\_WH

Start Time

8:15:17 PM

End Time

8:15:18 PM

Total Duration

957ms

Scanned Bytes

762.2KB

Rows

92.2K

Query ID

018c2403-00e1-ad9c-0000-1f89000178f6

Session ID

529133577

SQL Text

1 update churn\_data set totalvisits=0;

Query Result

Results

row#	number of rows updated
------	------------------------

--data based on the Query-Id -Needs to be changed for your query

Query:

Create table restored\_churn\_data as

select \* from churn\_data before (statement =>'018c2403-00e1-ad9c-0000-1f89000178f6');

Output:

## Results Data Preview

✓ Query ID SQL 854ms 1 rows

Filter result...



Copy

Row	status ↓
1	Table RESTORED_CHURN_DATA successfully created.

--Sample data output

*select \* from restored\_churn\_data limit 10;*

Output:

Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	TOTALVISITS	LOCATION_NAME	POSTCODE	COUNTRY
1	2985458	27	M	Fit & Focused	732	Armley	LS13 2PZ	England
2	3014166	45	F	Fit & Focused	756	Bramley & Stanningley	LS28 6HZ	England
3	2387364	39	M	Fit & Focused	699	Mortlake and Barnes...	SW15 6LF	England
4	2445702	41	M	Fit & Focused	737	Pentwyn	CF23 9UL	Wales
5	926198	18	F	Happy Families	739	Bramley & Stanningley	LS13 2SU	England
6	2408910	55	M	Fit & Focused	727	Carfax	OX1 1AN	UK
7	2966749	4	M	Fit & Focused	730	Kirkstall	LS5 3QE	England
8	2941788	29	M	Comfy & Complacent	737	Picton	L7 6NG	England
9	753367	58	F	Fit & Focused	717	Castle	SA1 1AF	Wales
10	2521270	30	M	Fit & Focused	711	St James's	WC2R 3LA	England

## Clone table

We could have also used the clone table option to create the restored table. Zero-copy cloning are pointer-based metadata copies of the data that can be manipulated freely without affecting the actual prod(loads).


Query:

--data based on the Query-Id -Needs to be changed for your query

*Create table restored\_churn\_data\_clone clone*


*churn\_data before (statement =>'(Query Id of the sql query which wrongly updated the data);*

Output:

Results Data Preview	
✓	Query_ID SQL 515ms 1 rows
Filter result...	
<div>  <div>Copy</div> </div>	
Row	status
1	Table RESTORED_CHURN_DATA_CLONE successfully created.

--Sample data output

```
select * from restored_churn_data_clone limit 10;
```

Results Data Preview		Open History						
✓	Query_ID SQL	188ms	10 rows					
Filter result...				<div>  <div>Copy</div> </div>	Columns ▾			
Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	TOTALVISITS	LOCATION_NAME	POSTCODE	COUNTRY
1	2024319	45	M	Healthy & Wealthy	735	Bryanston and Dor...	NW1 5EA	England
2	2547017	58	M	Health Neglecters	710	Haseley Brook	OX9 7JA	England
3	2811522	41	M	Healthy & Wealthy	699	Tyseley & Hay Mills	B25 8NN	England
4	2182904	51	F	Healthy & Wealthy	716	Northolt Mandeville	UB5 4RN	England
5	2605430	59	M	Healthy & Wealthy	733	East Finchley	N2 0LE	England
6	2766227	26	F	Healthy & Wealthy	730	Kingstanding	B44 8AP	England
7	2971411	3	M	Healthy & Wealthy	739	Fallowfield	M14 7HE	England
8	2699489	39	F	Healthy & Wealthy	716	Frognaal and Fitzjohns	NW3 6XE	England
9	2751372	46	F	Healthy & Wealthy	723	Chorlton Park	M21 7PN	England
10	2518982	45	F	Health Neglecters	720	Forth	EH4 4UU	Scotland

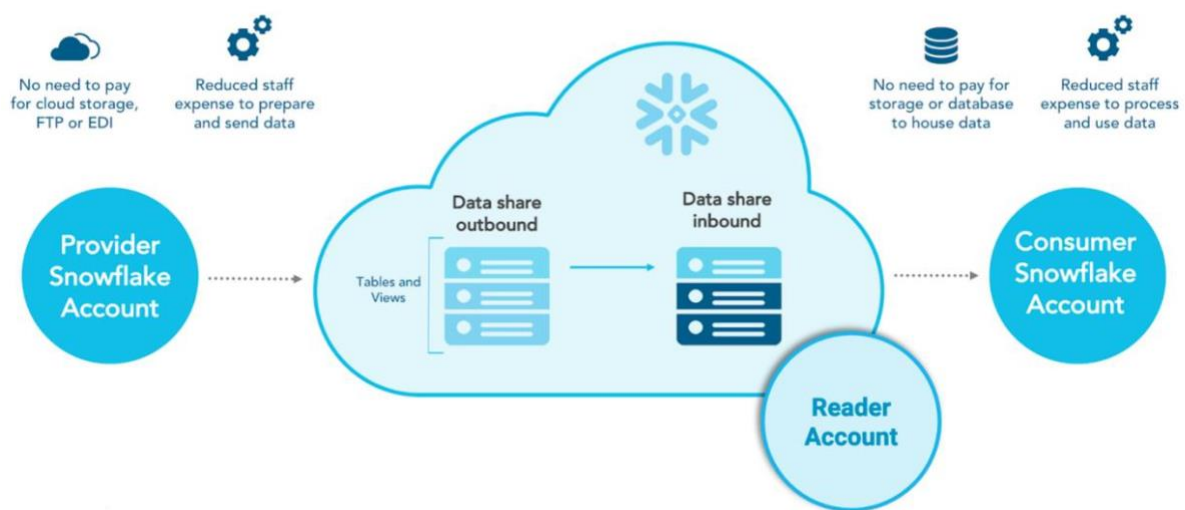
## 14. Data Sharing

### Data Sharing Understanding

Snowflake enables account-to-account sharing of data through *shares*, which are created by data providers and “imported” by data consumers, either through their own Snowflake account or a provisioned Snowflake Reader account.

With Data Sharing:

- There is only one copy of data, which lives in the data provider’s account.
- The shared data is always live, real-time and immediately available to consumers .
- Providers can establish revocable, fine-grained access grants to shares .



### Create outbound share

From the worksheet create a datashare called “Churn”.

Query:

```
create or replace share churn;
```

Output:

**Results** Data Preview✓ Query\_ID SQL 81ms  1 rows

Filter result...



Copy

Row	status
1	Share CHURN successfully created.

**Grant usage on CHURN\_DATA\_ANALYSIS database and schema to the share:**

From the worksheet provide grants for the table CHURN\_DATA to the share CHURN.

```
grant usage on database CHURN_DATA_ANALYSIS to share CHURN;  
grant usage on schema CHURN_DATA_ANALYSIS.RECENT TO SHARE CHURN;  
grant select on CHURN_DATA_ANALYSIS.RECENT.CHURN_DATA to share CHURN;
```

**Create a new reader account and add it to the share.**

Reader accounts (formerly known as “read-only accounts”) enable providers to share data with consumers who are not already Snowflake customers without requiring them to become Snowflake customers.

From the worksheet create the reader account

```
use role accountadmin;  
create managed account reader_acct1 admin_name = user1 , admin_password =  
'Br3x1tW734' , type = reader;
```

Output:

Note the account name and login url from the output. It will be used later to login to access the data share.

Results

Data Preview

✓

Query ID

SQL

800ms

1 rows

Filter result...

Copy

Row	status ↓
1	{ "accountName": "UF11136", "loginUrl": "https://uf11136.eu-west-1.snowflakecomputing.com" }

From the worksheet add the reader account to the data share CHURN.

*alter share churn add account=<Account name of the reader account from above query output>;*

297

298 `alter share churn add account=UF11136;`

299

Results

Data Preview

✓


Query\_ID

SQL

64ms

1 rows

Filter result...



Row	status
1	Statement executed successfully.

Go to the Shares tab and click Outbound to verify that your share has been created:

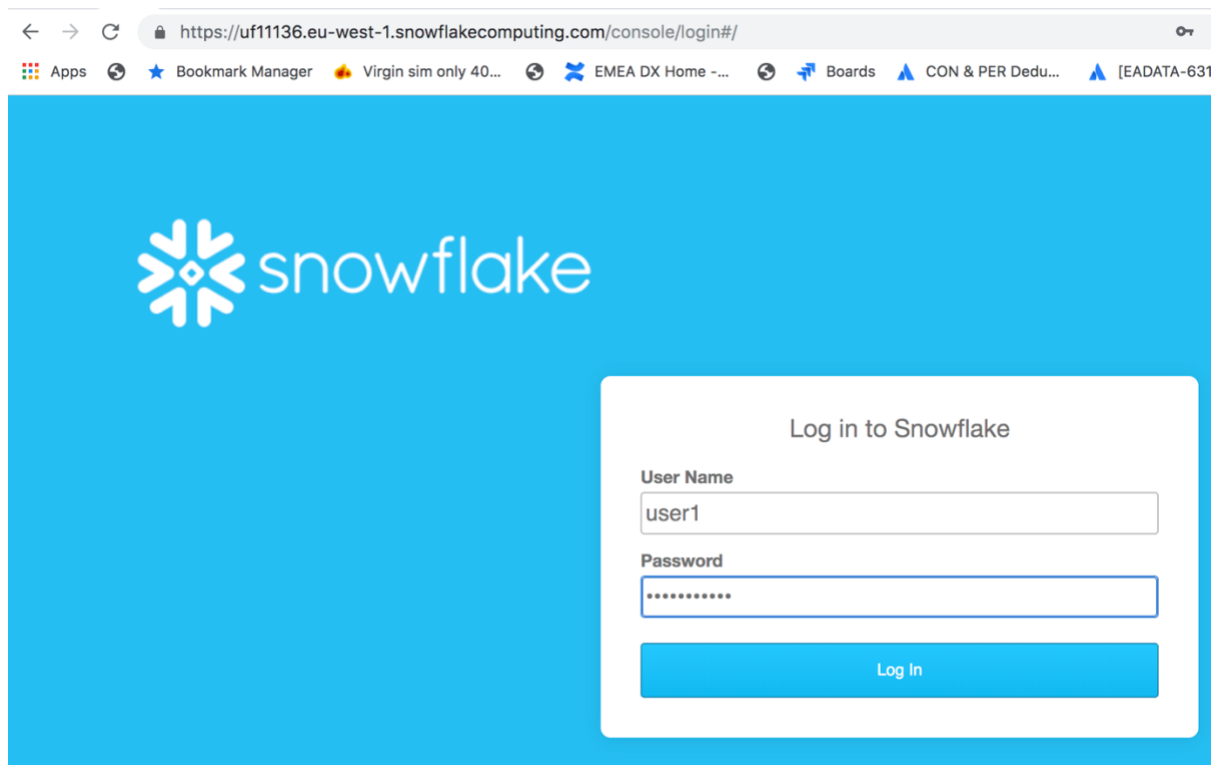
Shares					
Inbound	Outbound	+ Create Share	+ Add Consumers	✗ Edit	🗑 Drop
Search Outbound Shares		4 Outbound Shares		Columns ▾	
Share Name	Shared With	Database	↓ Creation Time	Owner	Comment
CHURN	UF11136	CHURN_DATA_ANALY...	21:03:19	ACCOUNTADMIN	

## Import the share from the Consumer account (Reader Account)

Open a new browser tab and navigate to the consumer account (defined in the login url of the output of the reader account creation).


Note: It will take a few minutes for the consumer account to be created.

Results		Data Preview	
✓	Query ID	SQL	800ms 1 rows
Filter result...		Download	Copy
Row	status ↓		
1	{"accountName":"UF11136","loginUrl":"https://uf11136.eu-west-1.snowflakecomputing.com"}		



← → ↻ https://uf11136.eu-west-1.snowflakecomputing.com/console/login#/ 🔑

Apps Bookmark Manager Virgin sim only 40... EMEA DX Home -... Boards CON & PER Dedu... [EADATA-631

 snowflake

Log in to Snowflake

User Name

Password

Log In

A window will pop up. Click on Dismiss and close the window.

From the worksheet, please run the below query to create the database from the share CHURN & provide grant to the ACCOUNTADMIN & SYSADMIN.

*use role accountadmin;*

*CREATE DATABASE "CHURN\_INPUT" FROM SHARE (Name of the parent snowflake account which created the share)."CHURN";*

*GRANT IMPORTED PRIVILEGES ON DATABASE "CHURN\_INPUT" TO ROLE "ACCOUNTADMIN";*

*GRANT IMPORTED PRIVILEGES ON DATABASE "CHURN\_INPUT" TO ROLE "SYSADMIN";*

From the worksheet, access the data share of the table CHURN\_DATA.

*select \* from "CHURN\_INP"."RECENT"."CHURN\_DATA";*

Output is as shown below:

Results Data Preview Open History

✓ Query ID SQL 117ms 10 rows

Filter result... Download Copy Columns

Row	CUSTOMERKEY	AGE	GENDER	SEGMENT	TOTALVISITS	LOCATION_NAME	POSTCODE	COUNTRY
1	2985458	27	M	Fit & Focused	0	Armley	LS13 2PZ	England
2	3014166	45	F	Fit & Focused	0	Bramley & Stannin...	LS28 6HZ	England
3	2387364	39	M	Fit & Focused	0	Mortlake and Barne...	SW15 6LF	England
4	2445702	41	M	Fit & Focused	0	Pentwyn	CF23 9UL	Wales
5	926198	18	F	Happy Families	0	Bramley & Stannin...	LS13 2SU	England
6	2408910	55	M	Fit & Focused	0	Carfax	OX1 1AN	UK
7	2966749	4	M	Fit & Focused	0	Kirkstall	LS5 3QE	England
8	2941788	29	M	Comfy & Complacent	0	Picton	L7 6NG	England
9	753367	58	F	Fit & Focused	0	Castle	SA1 1AF	Wales
10	2521270	30	M	Fit & Focused	0	St James's	WC2R 3LA	England



## 15. Appendix A

Snowflake's approach to access control combines aspects from both of the following models:

**Discretionary Access Control (DAC):** Each object has an owner, who can in turn grant access to that object.

**Role-based Access Control (RBAC):** Access privileges are assigned to roles, which are in turn assigned to users.

The key concepts to understanding access control in Snowflake are:

**Securable object:** An entity to which access can be granted. Unless allowed by a grant, access will be denied.

**Role:** An entity to which privileges can be granted. Roles are in turn assigned to users. Note that roles can also be assigned to other roles, creating a role hierarchy.

**Privilege:** A defined level of access to an object. Multiple distinct privileges may be used to control the granularity of access granted.

**User:** A user identity recognised by Snowflake, whether associated with a person or programme.

System-defined roles:

ACCOUNTADMIN:	(aka Account Administrator) Role that encapsulates the SYSADMIN and SECURITYADMIN system-defined roles. It is the top-level role in the system and should be granted only to a limited/ controlled number of users in your account.
SECURITYADMIN:	(aka Security Administrator) Role that can create, monitor, and manage users and roles. More specifically, this role is used to: Create users and roles in your account (and grant those privileges to other roles). Modify and monitor any user, role, or session. Modify any grant, including revoking it.
SYSADMIN:	(aka System Administrator) Role that has privileges to create warehouses and databases (and other objects) in an account. If, as recommended, you create a role hierarchy that ultimately assigns all custom roles to the SYSADMIN role, this role also has the ability to grant privileges on warehouses, databases, and other objects to other roles.
PUBLIC:	Pseudo-role that is automatically granted to every user and every role in your account. The PUBLIC role can own securable objects,

	<p>just like any other role; however, the objects owned by the role are, by definition, available to every other user and role in your account.</p> <p>This role is typically used in cases where explicit access control is not needed and all users are viewed as equal with regard to their access rights.</p>
--	---

Also we can create custom roles and assign privileges for any entities.