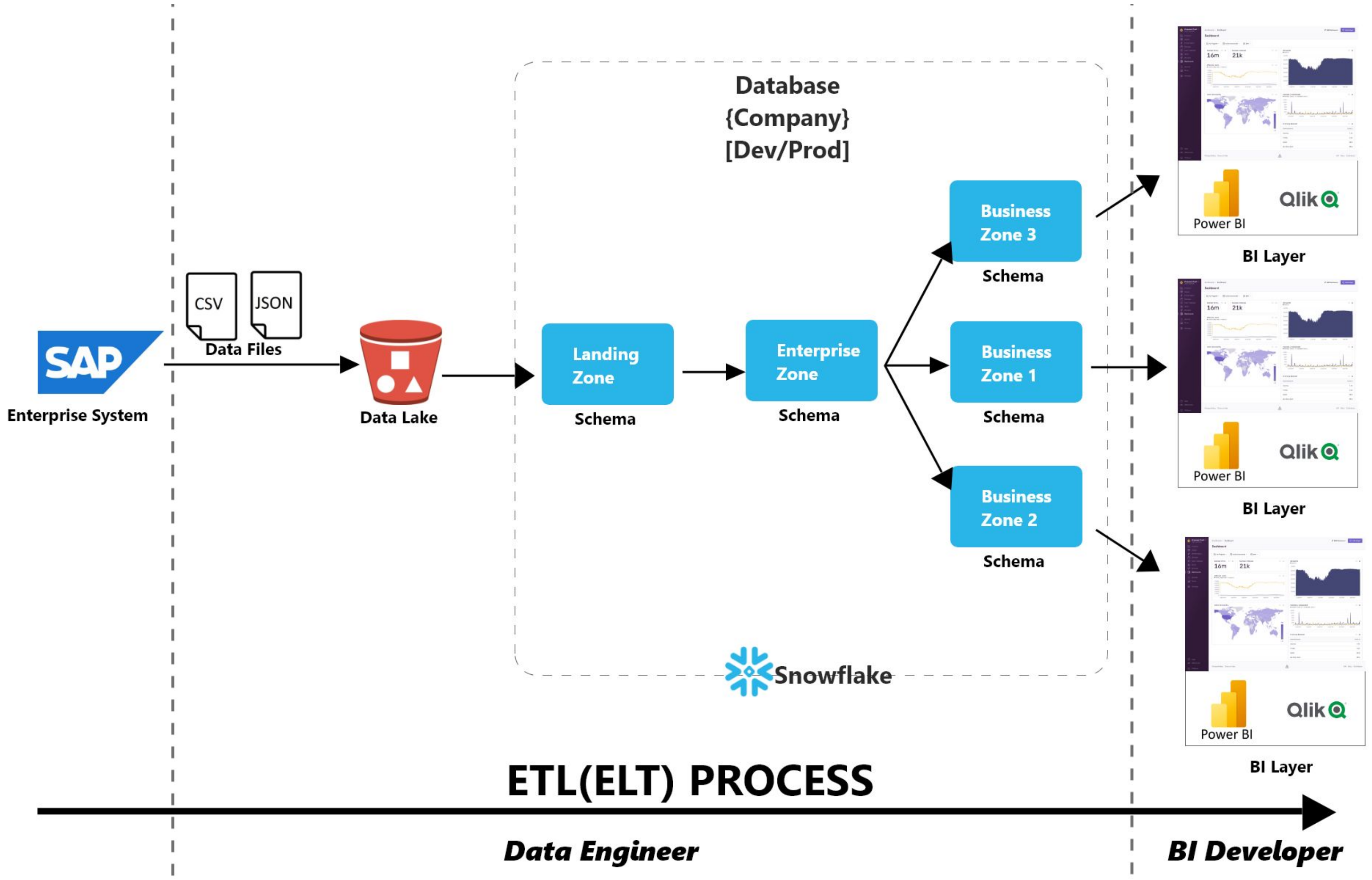
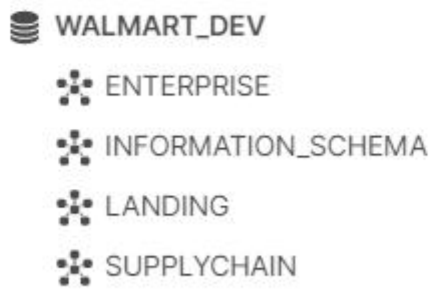
Snowflake

Agenda.

Snowflake Usage
Business Scenarios

Dataflow
Snowflake used as a Data warehouse in a BI platform.




WALMART_DEV
- ENTERPRISE
- INFORMATION_SCHEMA
- LANDING
- SUPPLYCHAIN

WeCloudData

Agenda.

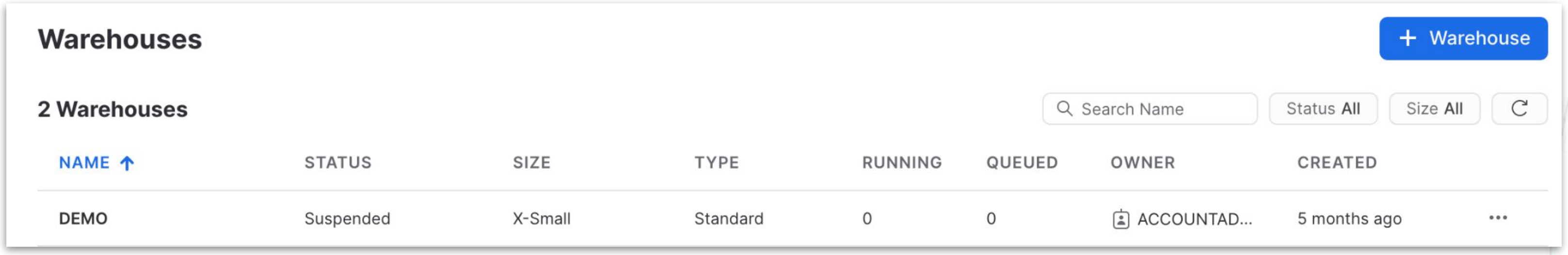
Components in Snowflake

Components

- **Database**
  - Database is an organization's data sets. Dev and Prod use different databases.
- **Schema**
  - schema shows a logical view of an database. It constructs views and tables. It represent a set of data in certain purpose.
- **Table and views**
  - Contain structured data
- **Worksheet**
  - Editor to write and run SQL.
- **Warehouse**
  - Engine to run queries.
- **Roles**
  - Access and operation permission group.
- **History**
  - Historical queries.

Warehouse
Components

❖ **Create Warehouse in Console**



Warehouses

+ Warehouse

2 Warehouses

Search Name | Status All | Size All

NAME ↑	STATUS	SIZE	TYPE	RUNNING	QUEUED	OWNER	CREATED
DEMO	Suspended	X-Small	Standard	0	0	ACCOUNTAD...	5 months ago

❖ **Create Warehouse in Query**

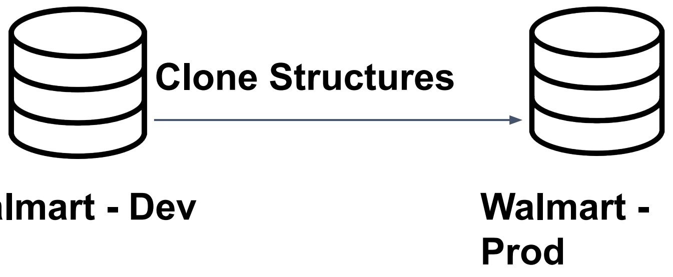
```
CREATE [ OR REPLACE ] WAREHOUSE [ IF NOT EXISTS ] <name>
WITH WAREHOUSE_SIZE = XSMALL | SMALL | MEDIUM | LARGE | XLARGE | XXLARGE | XXXLARGE |
X4LARGE | X5LARGE | X6LARGE
MAX_CLUSTER_COUNT = <num>
MIN_CLUSTER_COUNT = <num>
…..
;

USE WAREHOUSE <name>;
```

Useful Link: https://docs.snowflake.com/en/sql-reference/sql/create-warehouse.html

Try in system

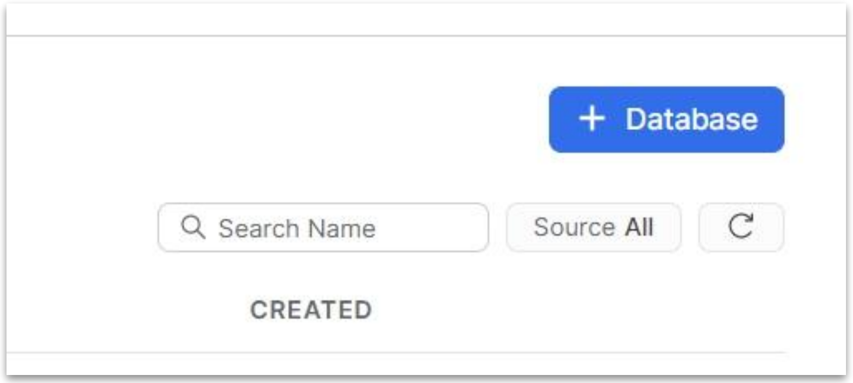
Database
Components

Definition: Database is an organization's data sets. Dev and Prod use different databases.



Creation

- Console
  - Creation



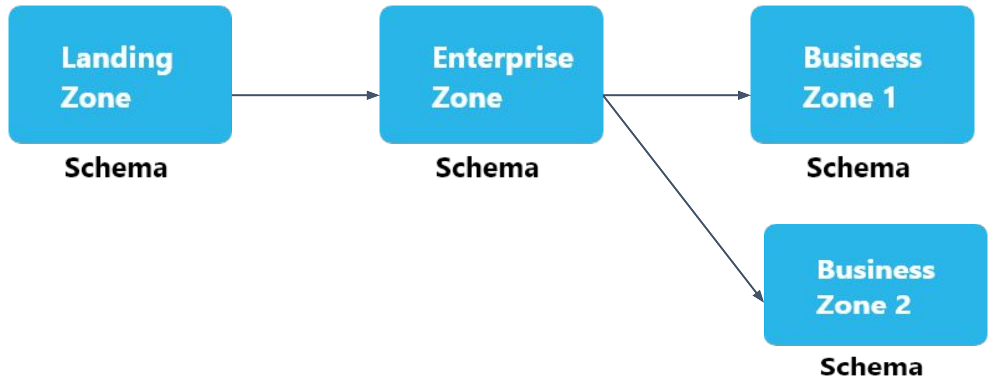
- Query

```
CREATE  DATABASE IF NOT EXISTS DEMO_DB;

USE DATABASE DEMO_DB;
```

Try in system

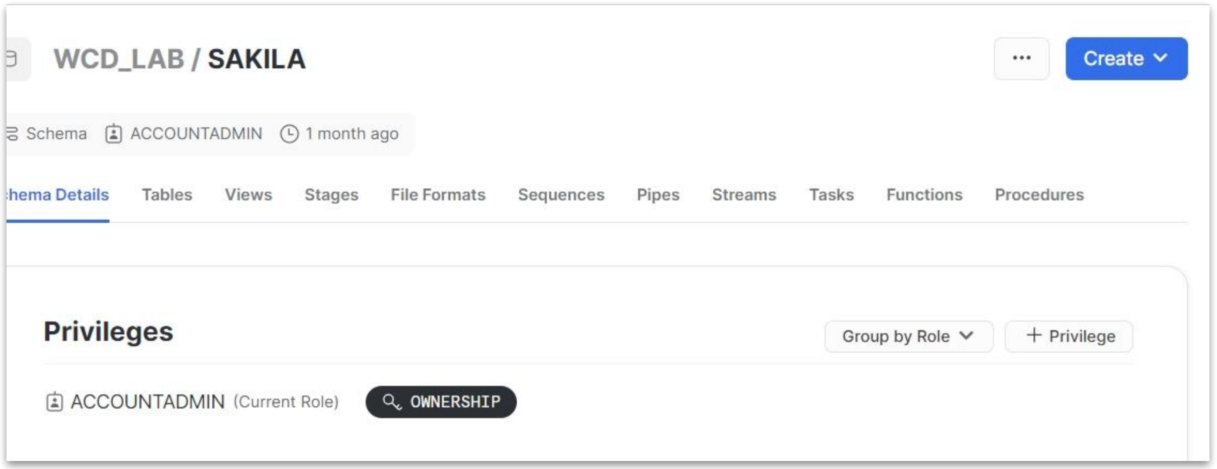
Schema
Components

Definition: Where you split your database logically.



Creation

- Console
  - Creation



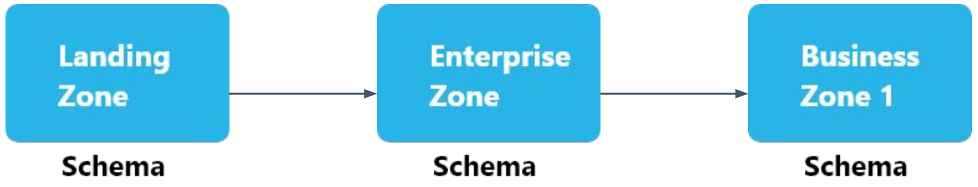
- Query

```
CREATE SCHEMA IF NOT EXISTS DEMO_DB.SCHMA;
```

Try in system

Schema
Components

Data Transit between schemas:



- **Create**

```
CREATE TABLE IF NOT EXISTS enterprise.table
AS (SELECT col_1, col_2 FROM landing.table);
```

- **Insert**

```
INSERT INTO enterprise.table (col_1, col_2)
(SELECT col_1, col_2 FROM landing.table);
```

Try in system

Tables and Views

Components

Name	Definition	Usages
Regular Table	Regular Table is a collection that contains the data physically, which means the data take store space in hardwares.	Contain the formal tables in the database.
Transient Table	Transient tables are similar to regular tables with the main difference that they do not have a Fail-safe period.	It is mainly used as a staging table in ETL process.
Temporary Table	Temporary table is a physical table, but it only exist within the **session** in which they were created and persist only for the remainder of the session.They are not visible to other users or sessions.	Temp table is rarely used on ETL process. It is only used when you want to create some simple and temporary staging table in ETL; or when you need to do some testing but don't want impact the entire database. [Try in system]
View	A view allows the result of a query to be accessed as if it were a table.	It is used when you only need the query result, but don't need it contains data permanently. Mostly in ETL process.
Materialized View	A materialized view is a pre-computed data set derived from a query and stored for later use. Querying a materialized view is faster than a view.	It is expensive than a regular view, so it is used when: ● you need faster and significant processing; ● the result is small size data; ● The aggregates that take a long time to calculate.

Regular , Transient, Temporary Table

Components

- **Table Information**

  ❖ **Show all the tables in a schema**

```
SHOW TABLES IN <database>.<schema>;
```

  ❖ **Display a specific table information**

```
DESCRIBE TABLE <database>.<schema>.<table>;
```

  ❖ **Show the columns in a table**

```
SHOW COLUMNS IN <database>.<schema><table>;
```

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

Try in system

Regular , Transient, Temporary Table

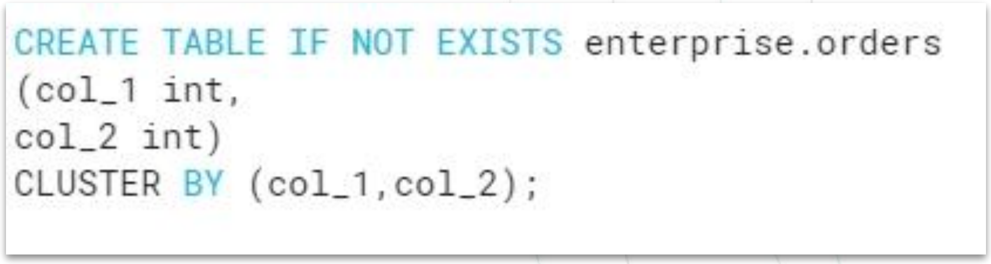
Components

- **Table Creation**
  - ❖ **Create an table**

```
CREATE TABLE [TRANSIENT TABLE, TEMP TABLE].....
```

  - ❖ **Create an empty table**

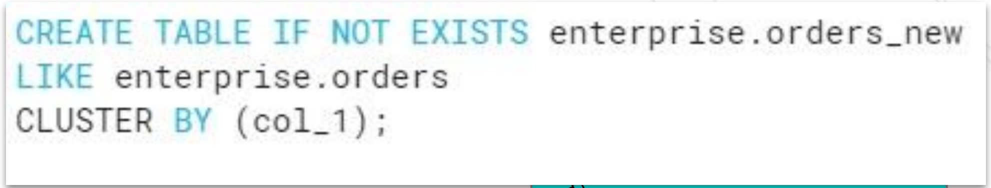
```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS]<table name>
(<col1_name> <col_type>,
 <col2_name> <col_type>,
 …….. 
 <coln_name> <col_type>)
[CLUSTER BY (col1_name,col2_name)];
```



```
CREATE TABLE IF NOT EXISTS enterprise.orders
(col_1 int,
col_2 int)
CLUSTER BY (col_1,col_2);
```

  - ❖ **Create an empty table by copying another table structure**

```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS]<table name>
LIKE <another table name>
[CLUSTER BY (col1_name,col2_name)];
```



```
CREATE TABLE IF NOT EXISTS enterprise.orders_new
LIKE enterprise.orders
CLUSTER BY (col_1);
```

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

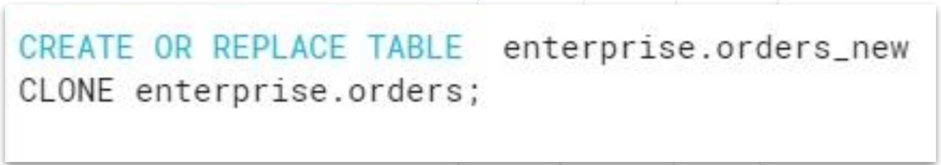
Try in system

Regular , Transient, Temporary Table
Components

- **Table Creation**

❖ **Create a table by clone another table totally**

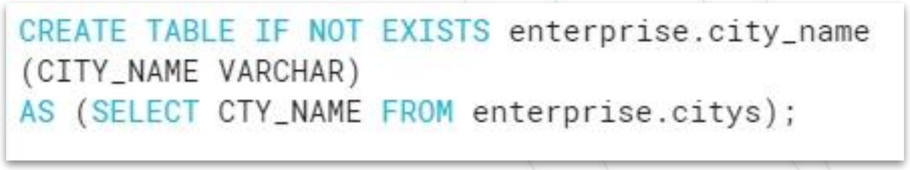
```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS]<table name>
CLONE <another table>
[CLUSTER BY (col1_name,col2_name)];
```



```
CREATE OR REPLACE TABLE  enterprise.orders_new
CLONE enterprise.orders;
```

❖ **Create an table by conditionally copying another table**

```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS]<table name>
(<col1_name> <col_type>,
 <col2_name> <col_type>,
 <col3_name> <col_type>)
AS
 (SELECT col1,col2,col1 FROM <another table>)
[CLUSTER BY (col1_name,col2_name)];
```



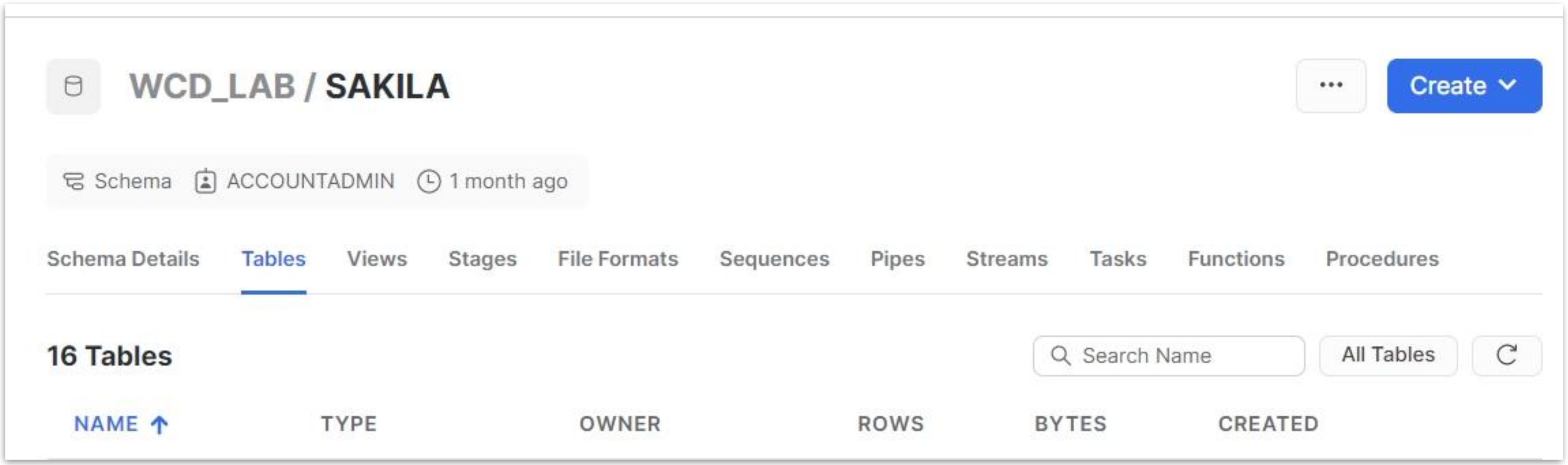
```
CREATE TABLE IF NOT EXISTS enterprise.city_name
(CITY_NAME VARCHAR)
AS (SELECT CTY_NAME FROM enterprise.citys);
```

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

Try in system

Regular , Transient, Temporary Table
Components

- **Table Creation**



WCD_LAB / SAKILA

Schema ACCOUNTADMIN 1 month ago

Schema Details | Tables | Views | Stages | File Formats | Sequences | Pipes | Streams | Tasks | Functions | Procedures

16 Tables

NAME ↑	TYPE	OWNER	ROWS	BYTES	CREATED

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

Try in system

Table
Components

- **Table Modification**

  ❖ **Change name**

  ```
  ALTER TABLE [IF EXISTS] <table name> RENAME TO <new table name>;
  ```

  ❖ **Add and change Column**

  ```
  ALTER TABLE [IF EXISTS] <table name> ADD COLUMN <col_name> <col_type>;
  ALTER TABLE [IF EXISTS] <table name> RENAME COLUMN <col_name> to <new col_name>;
  ALTER TABLE [IF EXISTS] <table name> ALTER COLUMN <col_name> SET TYPE <new data_type>;
  ```

- **Drop Table**

  ```
  DROP TABLE [IF EXISTS] <table name> ;

  UNDROP TABLE [IF EXISTS] <table name> ;
  ```

Useful Link: https://docs.snowflake.com/en/sql-reference/sql/alter-table.html

Try in system

View and Materialized view

Components

- **Table Information**

  ❖ **Show all the views in a schema**

```
SHOW VIEWS IN <database>.<schema>;
```

  ❖ **Display a specific view information**

```
DESCRIBE VIEW <database>.<schema>.<view>;
```

  ❖ **Show the columns in a view**

```
SHOW COLUMNS IN <database>.<schema><view>;
```

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

Try in system

View and Materialized Views

Components

- **View Creation**

  ❖ **Create an view by SELECT**

```
CREATE [OR REPLACE] [MATERIALIZED] VIEW [IF NOT
EXISTS]<view name>
(<col1_name> <col_type>,
 <col2_name> <col_type>,
 <col3_name> <col_type>)
AS
 (SELECT col1,col2,col1 FROM <another table>)
[CLUSTER BY (col1_name,col2_name)];
```

  ❖ **Drop View**

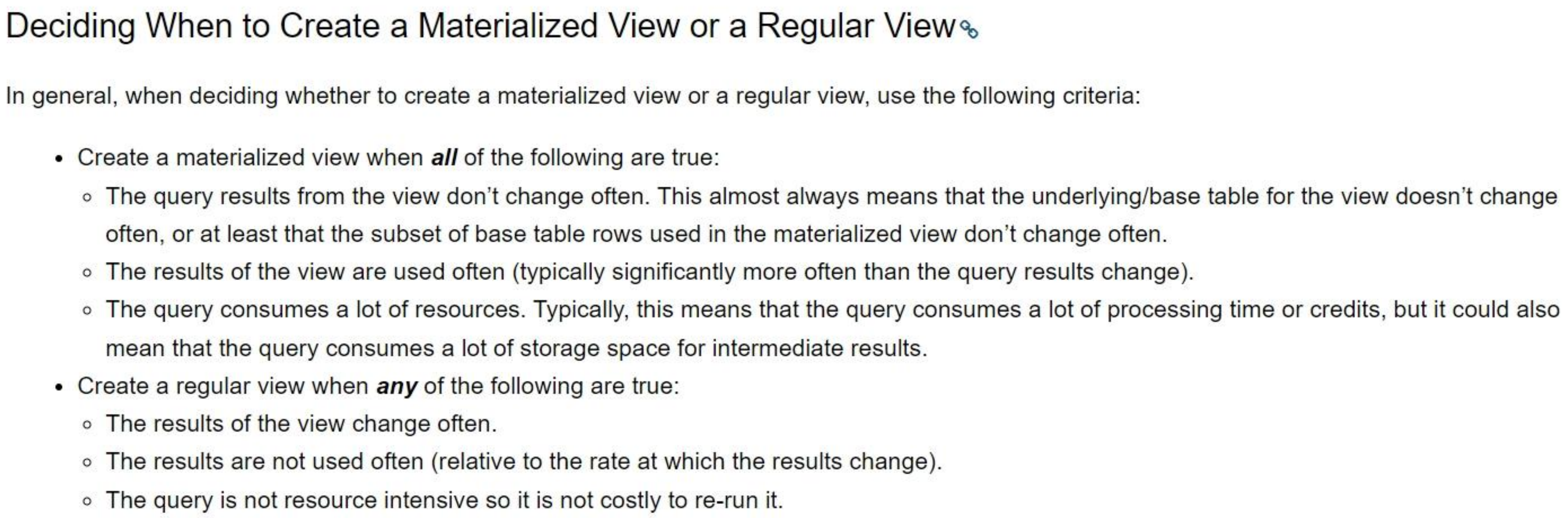
```
DROP [MATERIALIZED] VIEW [IF EXISTS] <view name> ;
```

Useful Link: https://docs.snowflake.com/en/sql-reference/ddl-table.html

Try in system

View and Materialized Views

Components



Deciding When to Create a Materialized View or a Regular View

In general, when deciding whether to create a materialized view or a regular view, use the following criteria:

- Create a materialized view when ***all*** of the following are true:
  - The query results from the view don't change often. This almost always means that the underlying/base table for the view doesn't change often, or at least that the subset of base table rows used in the materialized view don't change often.
  - The results of the view are used often (typically significantly more often than the query results change).
  - The query consumes a lot of resources. Typically, this means that the query consumes a lot of processing time or credits, but it could also mean that the query consumes a lot of storage space for intermediate results.
- Create a regular view when ***any*** of the following are true:
  - The results of the view change often.
  - The results are not used often (relative to the rate at which the results change).
  - The query is not resource intensive so it is not costly to re-run it.

Tables and Views Comparison

Components

Name	Speed	Cost
Cache	Very Fast	Very Expensive
Materialized View	Fast	Expensive
Regular Table	OK	OK
Regular View	Slow	Low

Try in system

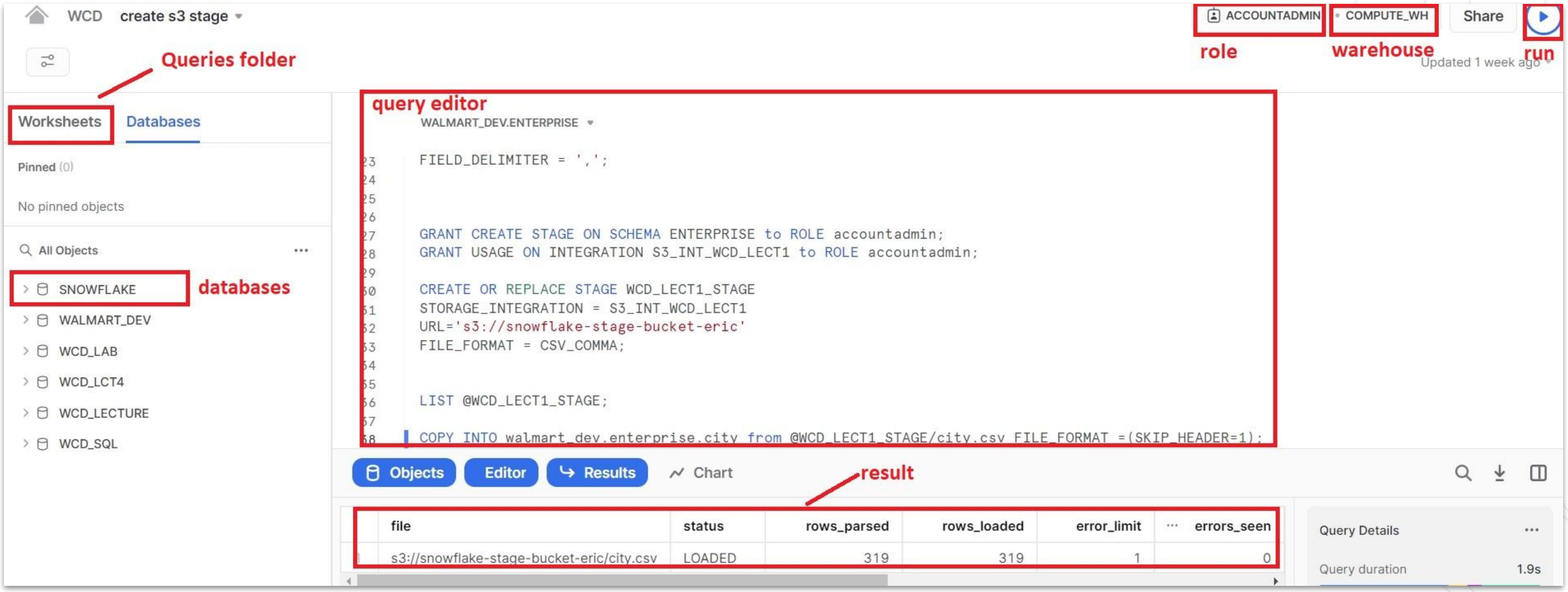
Tables – Data type

Components

Name	Definition	Example
INT	Integer	product_key, Is_flag (converted), qty (sometimes)
NUMERIC(38,2)	A number, system decide 0 ~ 2 digitals	sales, avg_xxx, sum_xxx
VARCHAR (XXX)	Characters: If you no number (xxx) is defined, system will adjust automatically; If defined, the length of character will be fixed. Usually don't define.	strings
BOOLEAN	Yes or No.	Is_flag
DATE	Date format: 2021-01-01	date
TIMESTAMP	Date and time: 2022-06-30 10:47:18.480 -0700	table update time

Try in system

Worksheet
Components



```
FIELD_DELIMITER = ',';



GRANT CREATE STAGE ON SCHEMA ENTERPRISE to ROLE accountadmin;
GRANT USAGE ON INTEGRATION S3_INT_WCD_LECT1 to ROLE accountadmin;

CREATE OR REPLACE STAGE WCD_LECT1_STAGE
STORAGE_INTEGRATION = S3_INT_WCD_LECT1
URL='s3://snowflake-stage-bucket-eric'
FILE_FORMAT = CSV_COMMA;


LIST @WCD_LECT1_STAGE;

COPY INTO walmart_dev.enterprise.city from @WCD_LECT1_STAGE/city.csv FILE_FORMAT =(SKIP_HEADER=1);
```

file	status	rows_parsed	rows_loaded	error_limit	errors_seen
s3://snowflake-stage-bucket-eric/city.csv	LOADED	319	319	1	0

Try in system

Role

Components

❖ **The purpose of the Role**

A Role can decide an account can use which:

- Database
- Schema
- Table(rarely happen)
- Warehouse

❖ **Create a Role**

```
CREATE [ OR REPLACE ] ROLE [ IF NOT EXISTS ] <role_name>;

GRANT USAGE DATABASE <database_name> TO ROLE <role_name>;
```

Try in system

Business Scenarios

Components

Know Snowflake

SnowSQL CLI

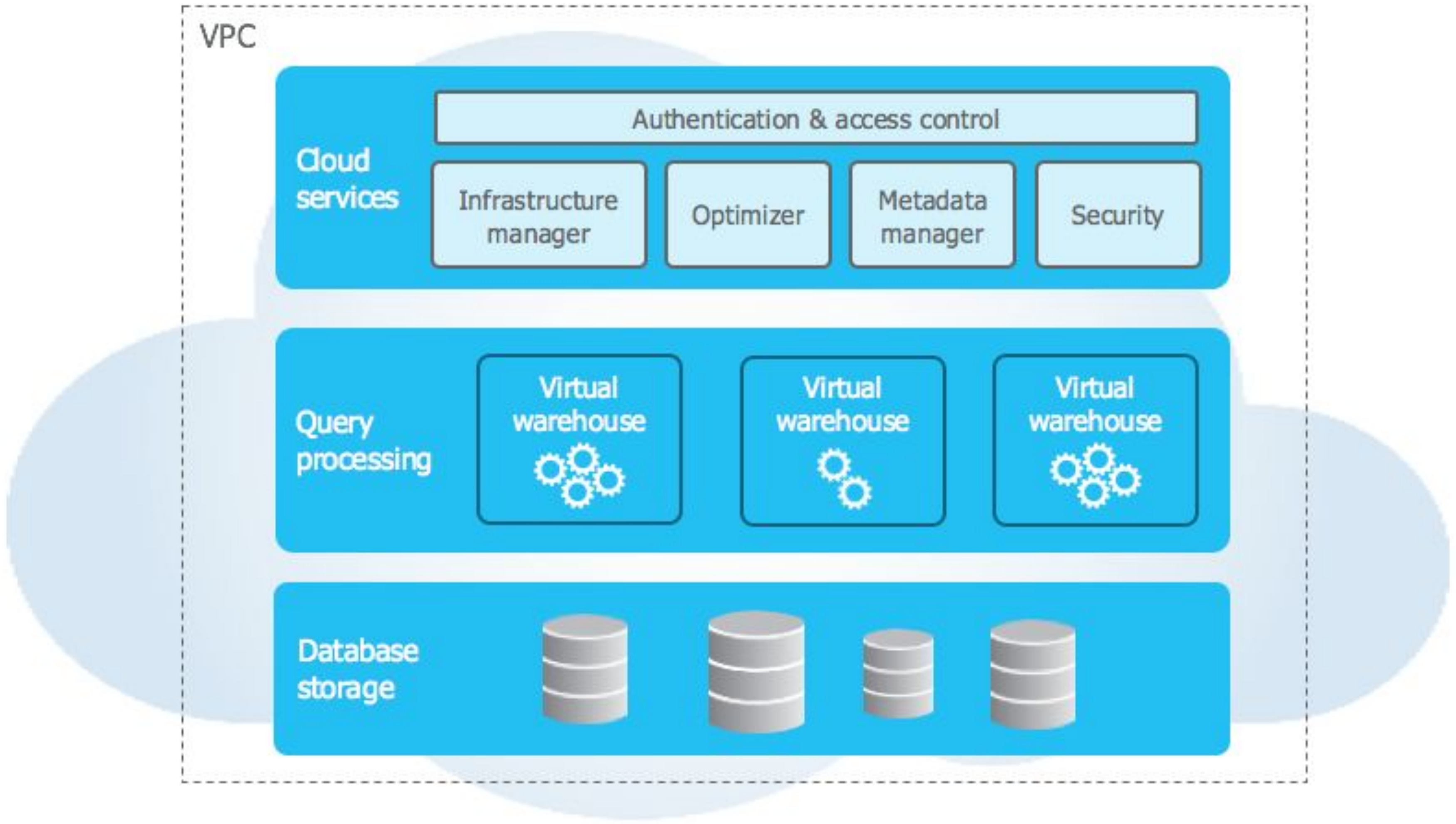
Loading Data

3rd Party Connections

Agenda.

Snowflake Architecture

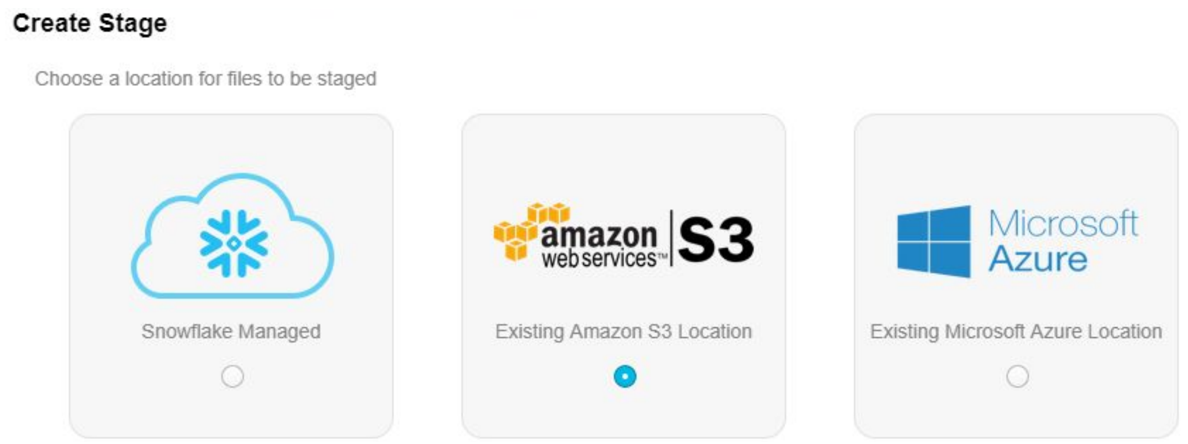
Know Snowflake



Snowflake Storage

Know Snowflake

❖ **The freedom to store your data**



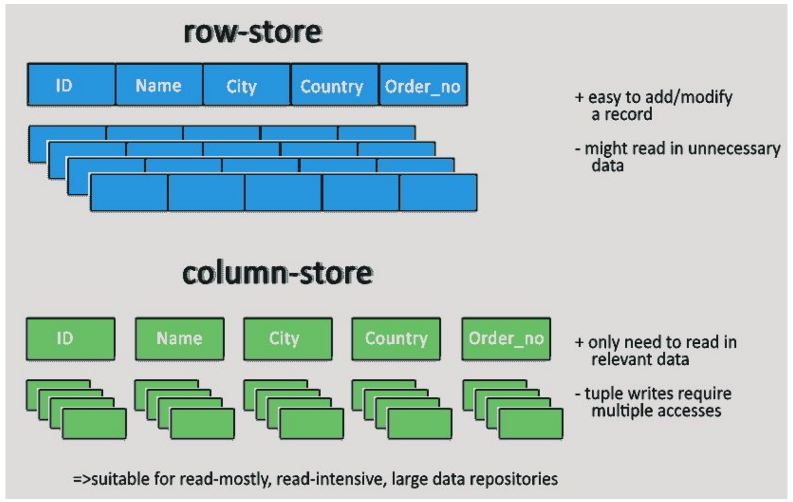
Create Stage

Choose a location for files to be staged

Snowflake Managed
amazon web services S3
Existing Amazon S3 Location
Microsoft Azure
Existing Microsoft Azure Location

Snowflake Managed ○ | [✓ Existing Amazon S3 Location] | Existing Microsoft Azure Location ○

❖ **The Data store in column**



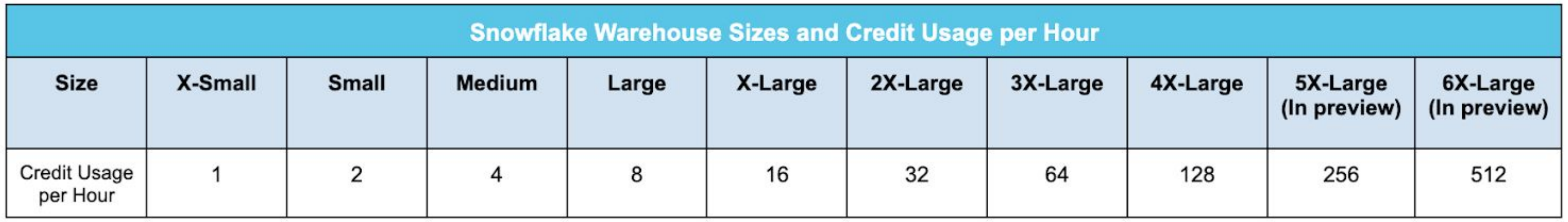
row-store
ID
Name
City
Country
Order_no
+ easy to add/modify a record
- might read in unnecessary data
column-store
ID
Name
City
Country
Order_no
+ only need to read in relevant data
- tuple writes require multiple accesses
=>suitable for read-mostly, read-intensive, large data repositories

- Good at where clause with keys
- Good at joining
- Primary key, foreign key will not useful in snowflake
- updating will be expensive
  - So avoid 'update'

Try in system

Snowflake Pricing
Know Snowflake

- **Storage**
  - $23 USD per compressed TB each month of data stored in US.
- **Computing**
  - Compute costs are $0.00056 **per second** for **each credit** consumed on Snowflake Standard Edition.



Snowflake Warehouse Sizes and Credit Usage per Hour

Size	X-Small	Small	Medium	Large	X-Large	2X-Large	3X-Large	4X-Large	5X-Large (In preview)	6X-Large (In preview)
Credit Usage per Hour	1	2	4	8	16	32	64	128	256	512

Example: *X-Large*
16 x 3600 x 0.00056 = 32.256 dollars

Agenda.

installation on Linux (For Ubuntu)

SnowSQL CLI

1. **Go to Download Folder**
   cd ~/Downloads

1. **Download the SnowSQL CLI Application**
   wget https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/linux_x86_64/snowflake-snowsql-1.2.21-1.x86_64.rpm

1. **Install RPM Package Manager**
   sudo apt-get update
   sudo apt-get install alien -y

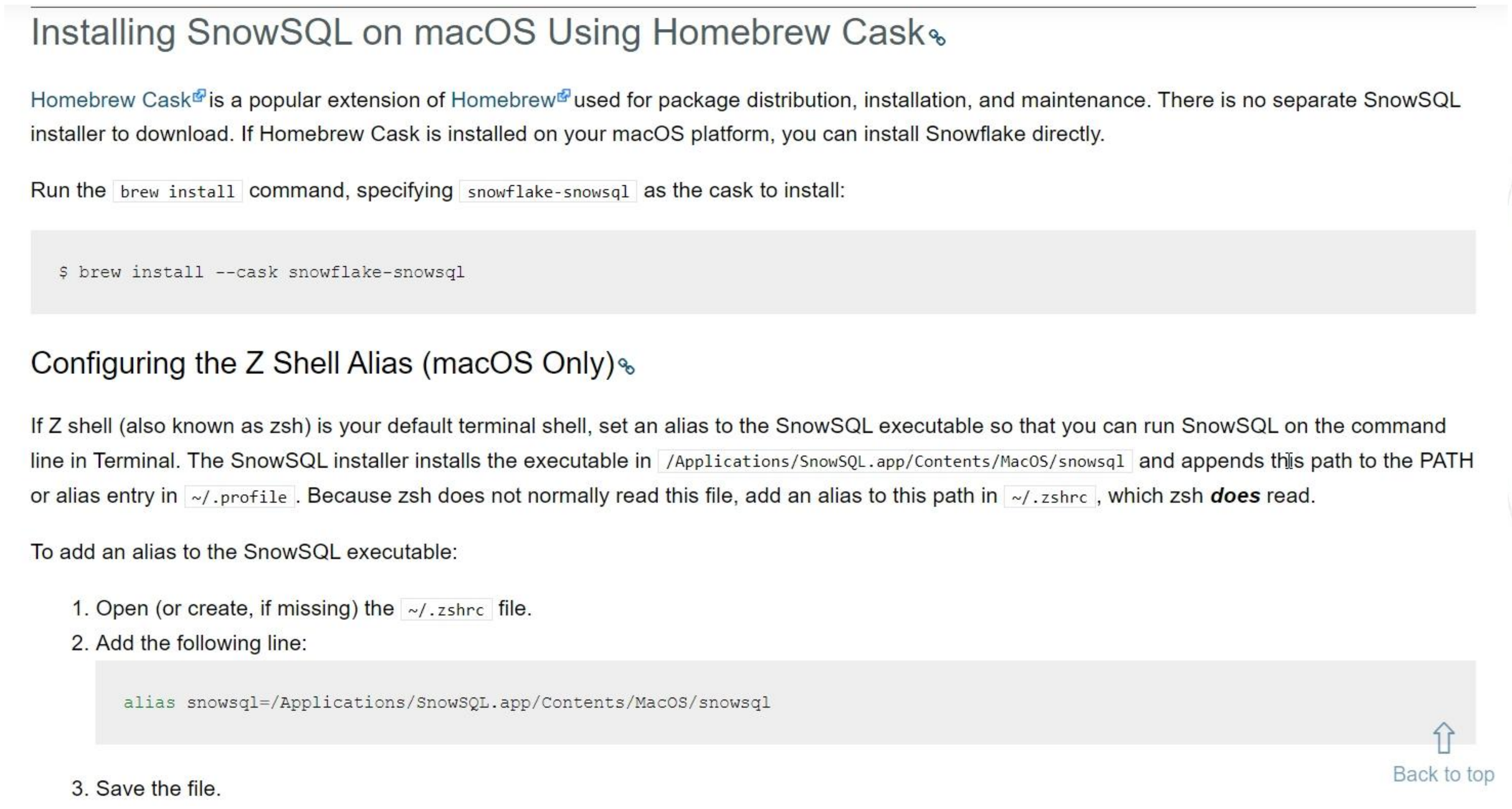
1. **Install the SnowSQL**
   sudo alien -i snowflake-snowsql-1.2.21-1.x86_64.rpm

1. **Check version**
   snowsql -v

Useful Link: https://docs.snowflake.com/en/user-guide/snowsql-install-config.html#installing-snowsql-on-linux-using-the-rpm-package

installation on Mac

SnowSQL CLI



Installing SnowSQL on macOS Using Homebrew Cask

Homebrew Cask is a popular extension of Homebrew used for package distribution, installation, and maintenance. There is no separate SnowSQL installer to download. If Homebrew Cask is installed on your macOS platform, you can install Snowflake directly.

Run the `brew install` command, specifying `snowflake-snowsql` as the cask to install:

```
$ brew install --cask snowflake-snowsql
```

Configuring the Z Shell Alias (macOS Only)

If Z shell (also known as zsh) is your default terminal shell, set an alias to the SnowSQL executable so that you can run SnowSQL on the command line in Terminal. The SnowSQL installer installs the executable in `/Applications/SnowSQL.app/Contents/MacOS/snowsql` and appends this path to the PATH or alias entry in `~/.profile`. Because zsh does not normally read this file, add an alias to this path in `~/.zshrc`, which zsh ***does*** read.

To add an alias to the SnowSQL executable:

1. Open (or create, if missing) the `~/.zshrc` file.
2. Add the following line:

   ```
   alias snowsql=/Applications/SnowSQL.app/Contents/MacOS/snowsql
   ```

3. Save the file.

Back to top

Useful Link: https://docs.snowflake.com/en/user-guide/snowsql-install-config.html#installing-snowsql-on-linux-using-the-rpm-package

WeCloudData

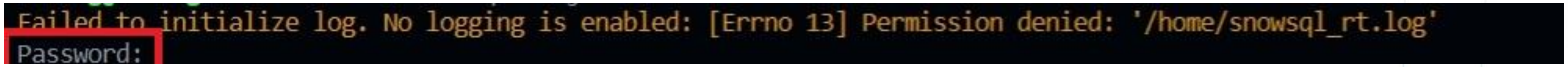
Connect SnowSQL with Snowflake

SnowSQL CLI

1. **Use command with account-name and username**

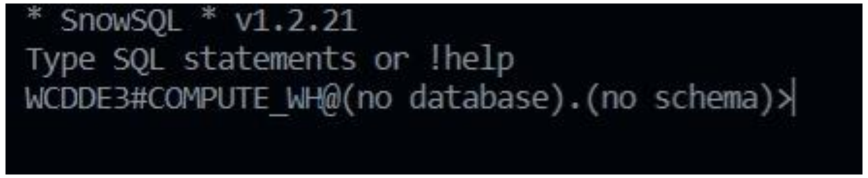
   snowsql -a <account-name> -u <username>

2. **Input Password**



```
Failed to initialize log. No logging is enabled: [Errno 13] Permission denied: '/home/snowsql_rt.log'
Password:
```

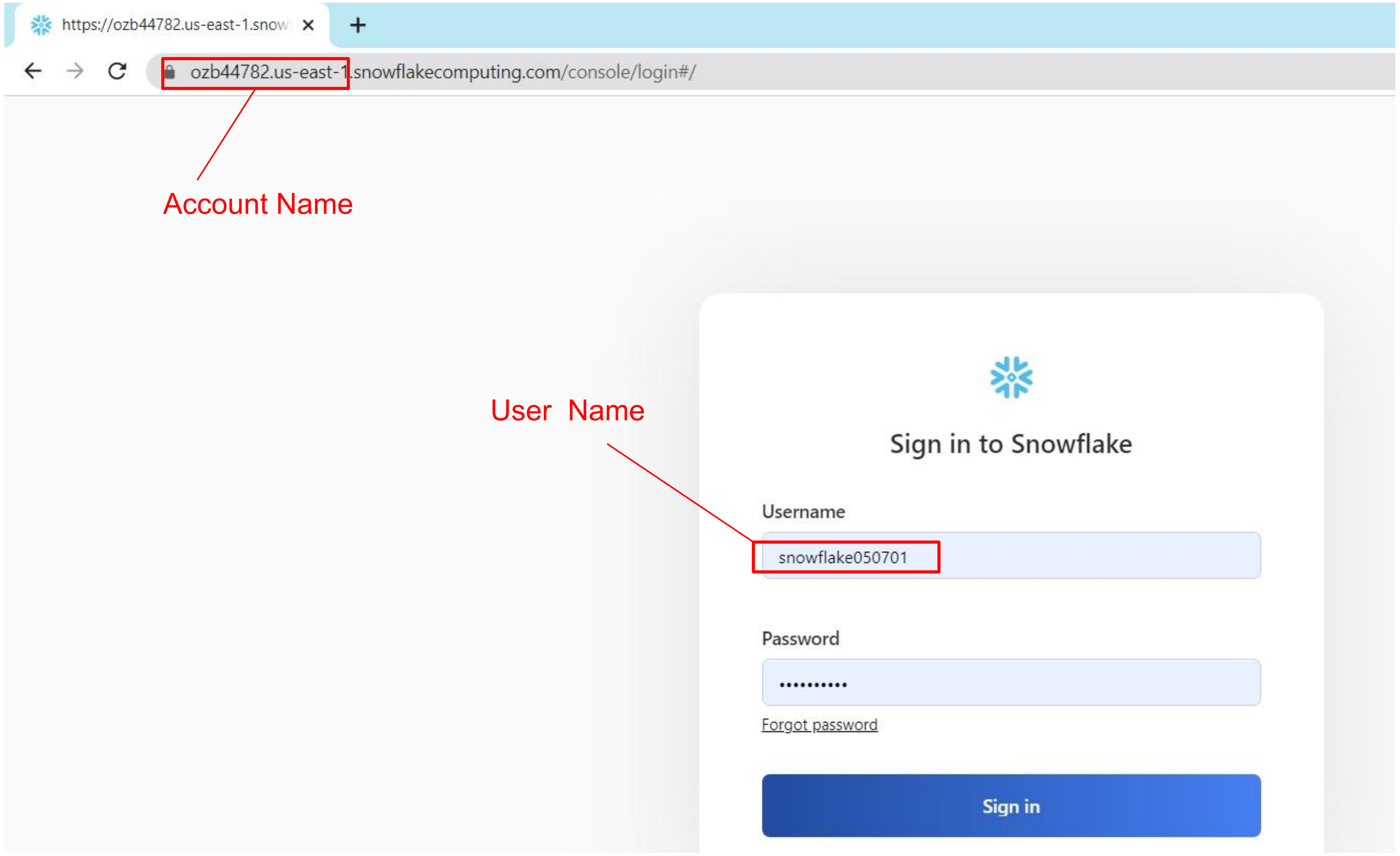
3. **SnowSQL start running**



```
* SnowSQL * v1.2.21
Type SQL statements or !help
WCDDE3#COMPUTE_WH@(no database).(no schema)>
```

Connect SnowSQL with Snowflake

SnowSQL CLI

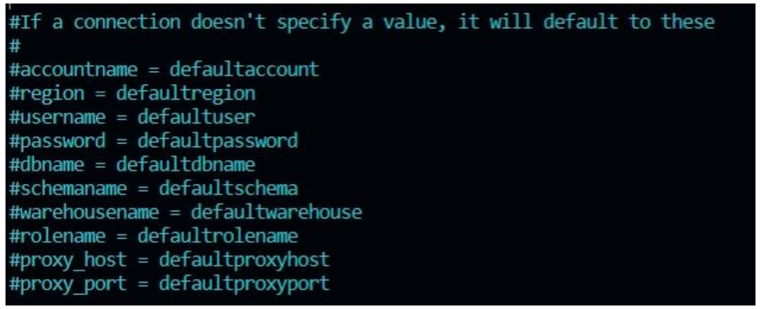


SnowSQL config
SnowSQL CLI

Why Config?

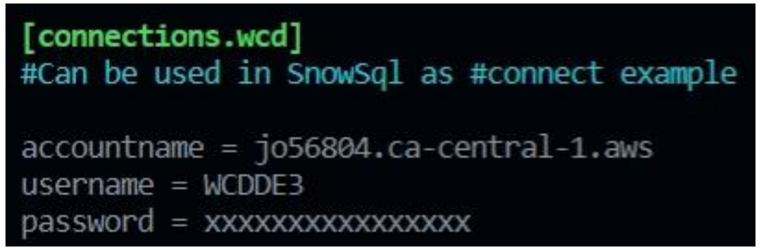
- **Connection**

➔ Default Connection Setting:



```
#If a connection doesn't specify a value, it will default to these
#
#accountname = defaultaccount
#region = defaultregion
#username = defaultuser
#password = defaultpassword
#dbname = defaultdbname
#schemaname = defaultschema
#warehousename = defaultwarehouse
#rolename = defaultrolename
#proxy_host = defaultproxyhost
#proxy_port = defaultproxyport
```

➔ Specific Connection Setting:



```
[connections.wcd]
#Can be used in SnowSql as #connect example

accountname = jo56804.ca-central-1.aws
username = WCDDE3
password = xxxxxxxxxxxxxxxx
```

Config file

Location: `~/.snowsql/config`

Open: *nano ~/.snowsql/config*

Connection:

default setting: *snowsql ………*

Specific setting: *snowsql -c wcd*

Try in system

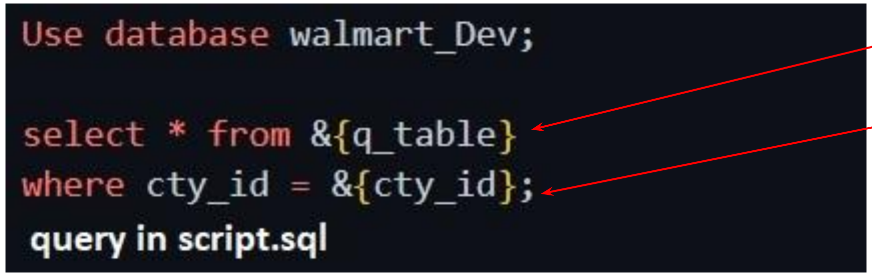
SnowSQL Command

SnowSQL CLI

➔ Options:

SnowSQL command

```
snowsql -c wcd -o variable_substitution=true -D q_table=enterprise.city -D cty_id=100001 -f script.sql
```



```
Use database walmart_Dev;

select * from &{q_table}
where cty_id = &{cty_id};
```

query in script.sql

-c Connection
-q Query
-o Option. Here must HAVE variable_substitution=true , otherwise, -D will not work.
-D Variables in script file
-f File of script

Try in system

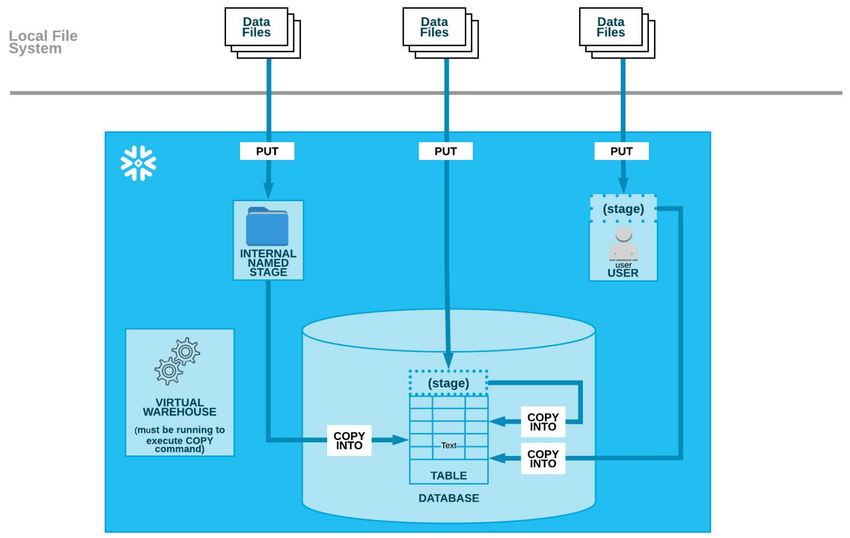
Useful Link: https://docs.snowflake.com/en/user-guide/snowsql-start.html#d-variable

Agenda.

Load Data with Query – Steps
Loading data

Step 1
Upload (i.e. **stage**) one or more data files to a Snowflake stage (named internal stage or table/user stage) using the **PUT** command.

Step 2
Use the **COPY INTO** <table> command to load the contents of the staged file(s) into a Snowflake database table.



Useful Link: https://docs.snowflake.com/en/user-guide/data-load-local-file-system.html

Staging Area

Loading data

Before a file being loaded into a table, it must be loaded into a Staging Area first. There are 2 important types of Staging Areas — User Stages and Named Stages:

User Stages

By default, each user and table in Snowflake is automatically allocated an internal stage for staging data files to be loaded. To list the file under the user stage use ‘*LIST @~;*’

Named Stages

Named stages are database objects that provide the greatest degree of flexibility for data loading.Named stages are optional but recommended when you plan regular data loads that could involve multiple users and/or tables. Usually it is stages integrate with other data lake, such as S3.

Useful Link: https://docs.snowflake.com/en/user-guide/data-load-local-file-system.html

Load Data From User Stages

Loading data

1. **Load data into Staging Area (***The process must be done in CLI***)**

```
snowsql -c wcd -q "PUT file://countries.csv @~"
```

@~: is the User stage.
file://cities.csv is the file dir in the local system.

1. **Copy Data from Staging area to the table (***The process works in both Editor or CLI***)**

```
COPY INTO schma.countries FROM @~/countries.csv [FILE_FORMAT = (TYPE = CSV FIELD_DELIMITER = ',' SKIP_HEADER = 1)];
COPY INTO schma.countries FROM (SELECT $1, $2 FROM@~/cities.csv);
```

@~: is the User stage
/cities.csv is the file dir in the stage.

Useful Link: https://docs.snowflake.com/en/user-guide/data-load-local-file-system.html

Load Data From Named Stages (S3 Integration)

Loading data

A stage integrated with S3 can upload or download a file from S3 directly. In order to make this, you need to set:

- IAM role in AWS which allows snowflake access to S3.
- Create a INTEGRATION at Snowflake.
- Create a STAGE with such INTEGRATION.
- The detail steps to create a stage can be found from this Introduction.

The lecture scripts can be found here.

Useful Link: https://hevodata.com/learn/snowflake-unload-to-s3/

Demo for local stage

Create a Database WALMART_DEV, a schema ENTERPRISE, a table CITY:

1. **Load data to local stage “@~”from file with “PUT”, and “COPY” the stage file into the city table.**
   a. Create database, schema, and table with this query;
   b. download CLI on your Linux server, and config the connection;
   c. Load city data(download from here) in local stage “@~”, and then copy from “@~” to city table with command:

      snowsql -c wcd -q "put file://city.csv @~; copy into walmart_dev.enterprise.city from @~/city.csv.gz FILE_FORMAT =(SKIP_HEADER=1);"

Demo for S3 stage

2. **Create a S3 stage, Load data from S3 stage to city table.**
    b. Create a bucket on S3 called “snowflake-stage-bucket-<your name>”
    c. Create a IAM policy with this script template, you need to change the bucket name. Give the policy name “snowflake-stage-bucket-policy” or other name you want.
    d. Create an IAM Role 'snowflake-stage-bucket-role' from 'AWS account' for ‘This account’:
        i. Require eternal ID <0000>;
        ii. attach policy 'snowflake-stage-bucket-policy';
        iii. copy the arn;
    e. create a S3 INTEGRATION, with the ROLE ARN with this query template on snowflake in Snowflake.
    f. run “*DESC INTEGRATION<your integration name>*” to get '**STORAGE_AWS_IAM_USER_ARN**' and '**STORAGE_AWS_EXTERNAL_ID**'.
    g. Past the 2 codes to AWS IAM Role, by going to Role -->'trust relationships' , and replace the JSON with this template.

Demo
Loading data

j. Go to snow flake create a FORMAT called CSV_COMMA with this query. This step is to tell Snowflake, we are going to use the format csv to load data.

k. grant stage and integration to schema with the following query:
   - *GRANT CREATE STAGE ON SCHEMA ENTERPRISE to ROLE accountadmin;*
   - *GRANT USAGE ON INTEGRATION S3_INT_WCD_LECT1 to ROLE accountadmin;*

j. Create the STAGE with this query:

```
CREATE OR REPLACE STAGE WCD_LECT1_STAGE
      STORAGE_INTEGRATION = S3_INT_WCD_LECT1
      URL='s3://your bucket name'
      FILE_FORMAT = CSV_COMMA;
```

k. Upload the city.csv file to the bucket;

l. Check if the file has been in stage *WCD_LECT1_STAGE* with query:

```
List @WCD_LECT1_STAGE;
```

m. Copy file to city table from stage @WCD_LECT1_STAGE with query:

```
 COPY INTO walmart_dev.enterprise.city from @WCD_LECT1_STAGE/city.csv
FILE_FORMAT =CSV_COMMA;
```

Business Scenarios

Components

Know Snowflake

SnowSQL CLI

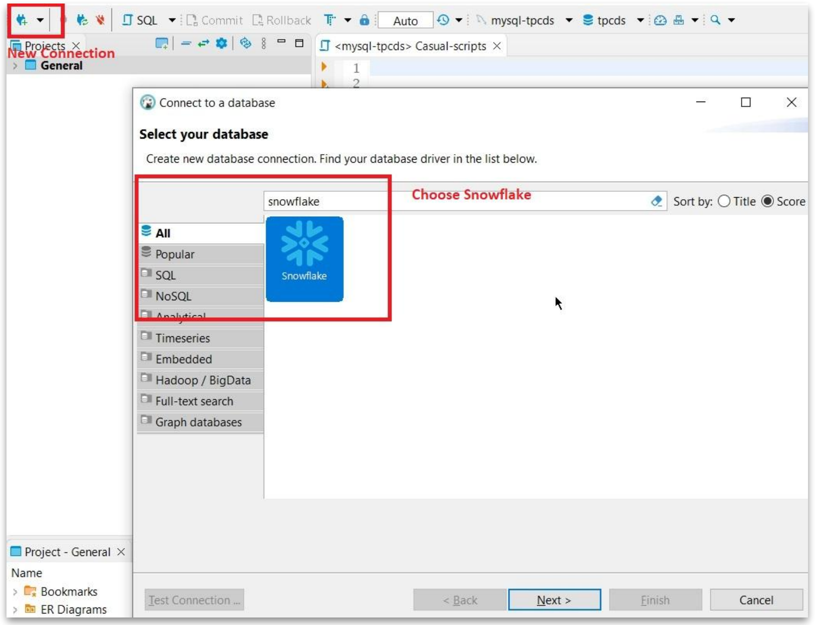
3rd Party Connections

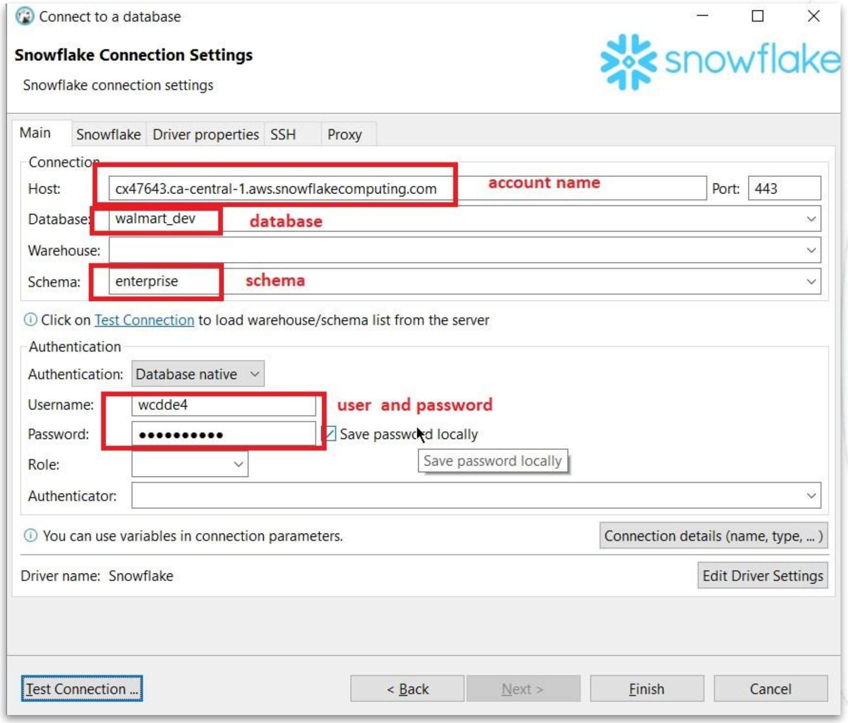
Agenda.

3rd Party tools Connection

Loading data

1. **DBeaver**




Connect to a database

Snowflake Connection Settings

Snowflake connection settings

Main | Snowflake | Driver properties | SSH | Proxy

Connection

Host:	cx47643.ca-central-1.aws.snowflakecomputing.com
Port:	443
Database:	walmart_dev
Warehouse:	
Schema:	enterprise

Click on Test Connection to load warehouse/schema list from the server

Authentication

Authentication:	Database native
Username:	wcdde4
Password:	••••••••••
Role:	
Authenticator:	

✓ Save password locally

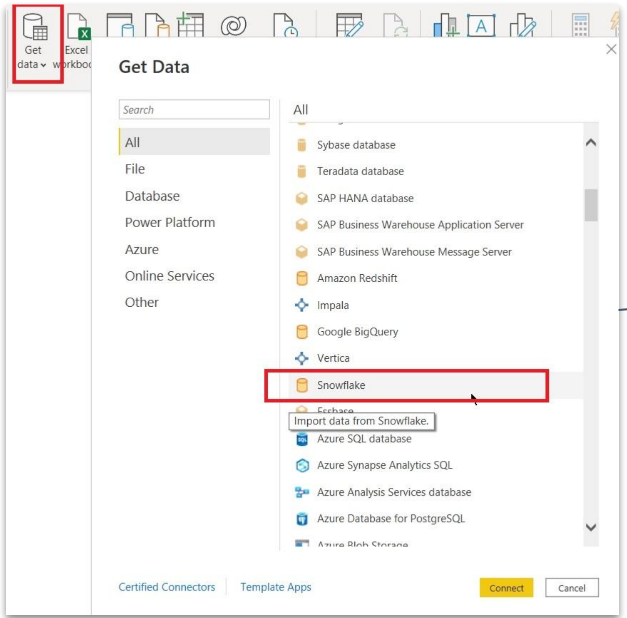
You can use variables in connection parameters.

Driver name: Snowflake

3rd Party tools Connection

Loading data

2. PowerBI



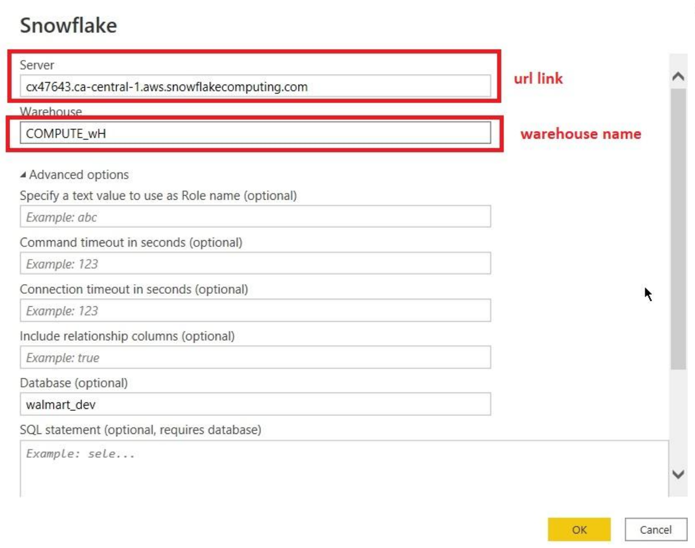
Get Data

Search

All
File
Database
Power Platform
Azure
Online Services
Other

All
Sybase database
Teradata database
SAP HANA database
SAP Business Warehouse Application Server
SAP Business Warehouse Message Server
Amazon Redshift
Impala
Google BigQuery
Vertica
Snowflake
Essbase
Import data from Snowflake.
Azure SQL database
Azure Synapse Analytics SQL
Azure Analysis Services database
Azure Database for PostgreSQL
Azure Blob Storage

Certified Connectors | Template Apps | Connect | Cancel



Snowflake

Server: cx47643.ca-central-1.aws.snowflakecomputing.com — url link

Warehouse: COMPUTE_wH — warehouse name

Advanced options

Specify a text value to use as Role name (optional): Example: abc

Command timeout in seconds (optional): Example: 123

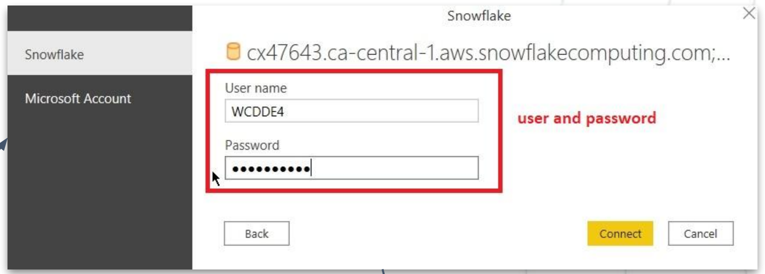
Connection timeout in seconds (optional): Example: 123

Include relationship columns (optional): Example: true

Database (optional): walmart_dev

SQL statement (optional, requires database): Example: sele...

OK | Cancel



Snowflake

Snowflake
Microsoft Account

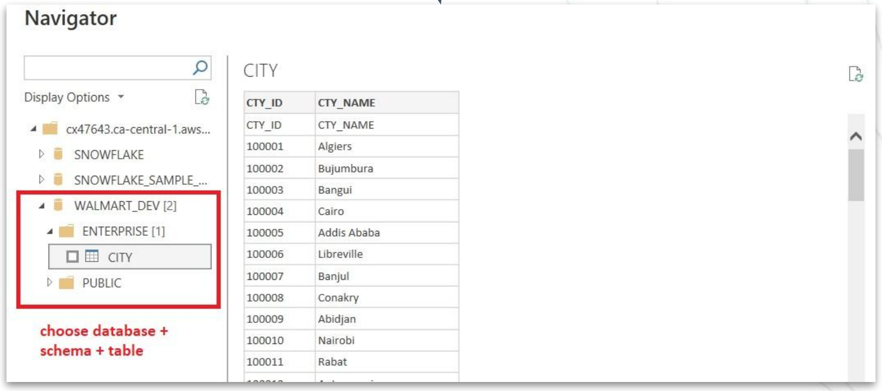
cx47643.ca-central-1.aws.snowflakecomputing.com;...

User name: WCDDE4

Password: ••••••••••

user and password

Back | Connect | Cancel



Navigator

Display Options

- cx47643.ca-central-1.aws...
  - SNOWFLAKE
  - SNOWFLAKE_SAMPLE_...
  - WALMART_DEV [2]
    - ENTERPRISE [1]
      - CITY
    - PUBLIC

choose database + schema + table

CITY

CTY_ID	CTY_NAME
CTY_ID	CTY_NAME
100001	Algiers
100002	Bujumbura
100003	Bangui
100004	Cairo
100005	Addis Ababa
100006	Libreville
100007	Banjul
100008	Conakry
100009	Abidjan
100010	Nairobi
100011	Rabat

WeCloudData

3rd Party tools Connection

Loading data

3. **Python**

pip install snowflake-connector-python

```
import snowflake.connector as sf
import pandas as pd

# make changes as per your credentials
user='snowflake050701'
password = 'Code123456'
account='ozb44782.us-east-1'
database='walmart_dev'
warehouse='COMPUTE_WH'
schema='enterprise'
role='accountadmin'

conn = sf.connect(user = user, password = password, \
                account = account, warehouse=warehouse, \
                 database=database,  schema=schema,  role=role)

def run_query(connection,query):
        cursor = conn.cursor()
        cursor.execute(query)
        cursor.close()

# sql = 'show warehouses;'
# run_query(conn, sql)

sql = 'select * from city;'
df = pd.read_sql(sql, conn)
df.head()
```