ETL Examples

-- Lab 01

-- Create a data base

```
CREATE DATABASE TIPS_SALES_PROD
COMMENT = 'This is topper tips sales production database';
```

-- Create Schema

```
CREATE SCHEMA "TIPS_SALES_PROD"."SALES"
COMMENT = 'This is TopperTips Sales Data';
```

-- Created a names internal stage

```
CREATE STAGE "TIPS_SALES_PROD"."SALES".SALES_STAGE
COMMENT = 'This is named internal SF manage stage';
```

--Creating a stage via clone (this operation may not work with WebUI)

```
CREATE STAGE TIPS_SALES_PROD.SALES.S3
CLONE USDA_NUTRIENT_STDREF.PUBLIC.S3_USDF_EXTERNAL_NAMED_STAGE
COMMENT = 'Cloning USDF S3 Bucket';
```

-- Creating a part table

```
CREATE TABLE TIPS_SALES_PROD.SALES.PART
CLONE SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.PART
COMMENT = 'This is cloned table from TPC AdHoc SF1';
```

-- <Error> SQL compilation error: Cannot clone from a table that was imported from a share.

-- alternative is -- use create like

```
CREATE TABLE TIPS_SALES_PROD.SALES.PART  LIKE SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.PART

COMMENT = 'This is create like  table from TPC AdHoc SF1';
```

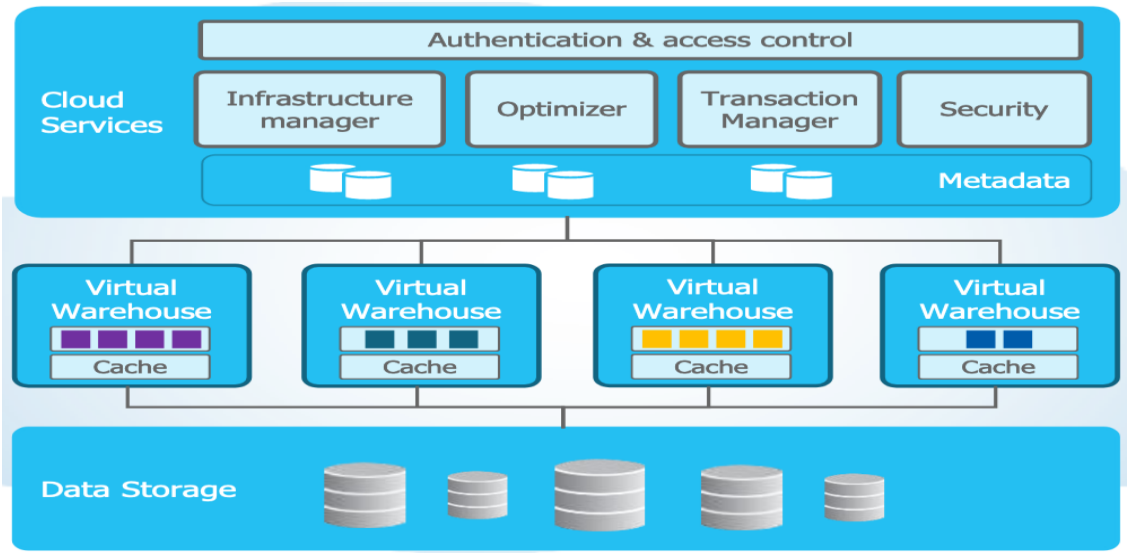
-- then insert

```
insert into TIPS_SALES_PROD.SALES.PART select * from
SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.PART
```

Three Layer

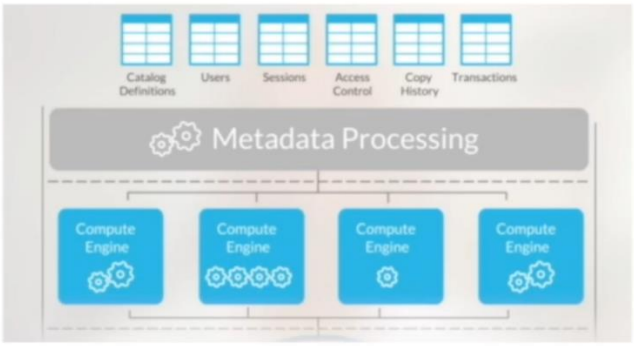
SNOWFLAKE ARCHITECTURE – CLOUD SERVICES

- Independent services managed entirely by Snowflake
  Scalable, cost-effective cloud data warehouse architecture
- Built-in Fault Tolerance and High Availability (Continuous Availability)
  Resilience against failures and delivers zero planned downtime
- Enterprise-grade Security
  Complete industry security standards compliance and best practices
- Metadata-driven Cost-based Optimizer
  Higher query performance through pruning and heuristic optimizations
- Time Travel and Zero-copy Cloning capabilities
  Time-based data restoration and near instantaneous data snapshots
- Data Sharing
  Secure, cross-account data access with absolutely no data movement
- Transaction Management with full ACID-compliance
  Guaranteed consistent views of data with high concurrency and performance



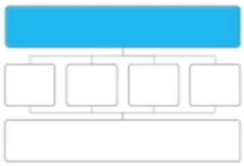
SNOWFLAKE METADATA SERVICE

- Provides foundation for:
  - Optimized micro-partition pruning
  - Time travel
  - Zero-copy cloning
  - Data sharing
- Linear Scaling Performance
  - Metadata processing is separate from compute resources

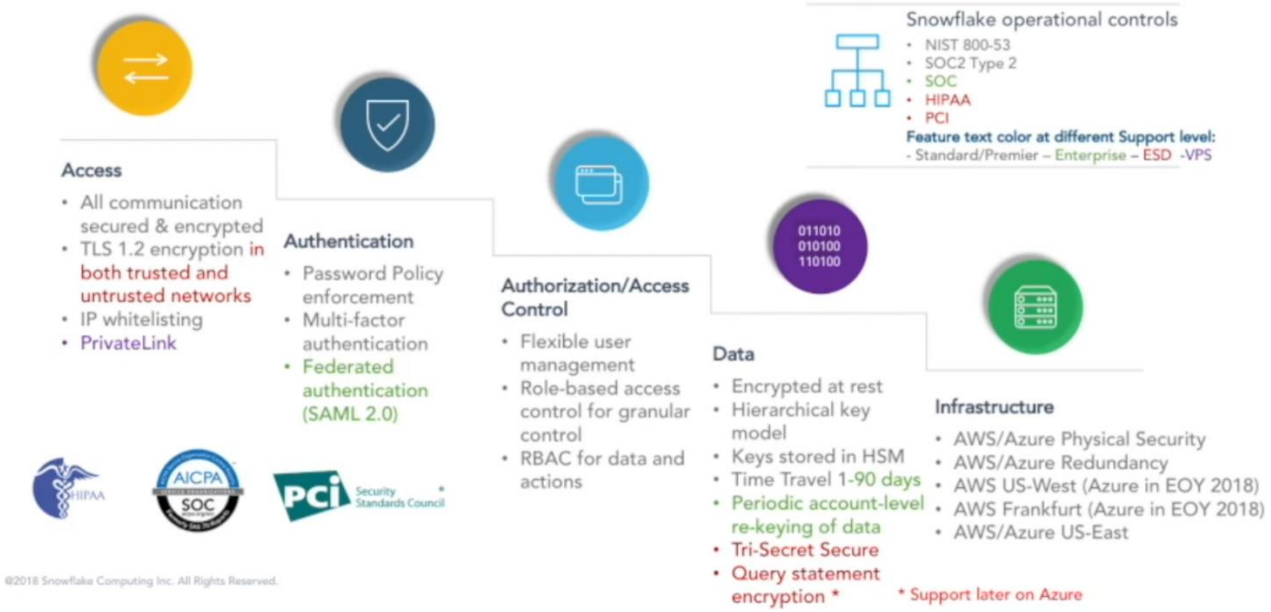


- Every aspect of Snowflake architecture leverages metadata
  - All user visible actions have metadata backing
    - catalog definitions, accounts, users, sessions, databases, schemas, tables, roles, grants, ...
  - User invisible metadata such as:
    - Statistics for query optimizer
    - Encryption keys, servers, …

SQL OPTIMIZER SERVICE



- SQL Optimizer [Cascades-style]
  - Top-down cost-based optimization (CBO)
- Automatic Statistics Gathering
  - On data loading and updates
- Built-in Join Order Optimization
  - No user input or tuning required
  - Use heuristics to automatically determine for more optimal join order for query performance
- Supports any and all data models
  - 3NF, snowflake, star, Data Vault
  - Optimizations for Star Schema (Bloom Filters, Broadcast Join)



Database Table View

-- ----------

1. No limit of db, db object in SF

2. default table is permanent table (temp, transient, external are other table types)

3. Non-permanent & transitory data can be stored in temporary & transient tables (short term data storage)

4. temporary & transient tables also called session specific data sets.

5. Temporary tables scope within the session in which they were created.

(so where the data is stored during the session)

- they don't support cloning

- not accessible via other sessions

- not recoverable (no fail safe) when session ends.

- SF also support temporary stages (other objects too)

- tmp data is part of storage cost and charges apply

- tmp table data can live more than 24hrs if session is alive and it cost storage charges.

- tmp table and permanent/transient table can have same name

- tmp table gets precedence over permanent & transient table

- this behaviour is given due to time travel (duplicate table name)

- tmp tables can not be converted to any other table once created

- retention period (time travel) is for 24 hours or the remainder of the session, whichever is shorter**.**

6. Transient Table

- There is a field in table (view) called is_transient

- Persist until explicitly dropped

- available to all users with privileges

- same as permanent table except fail safe

- contribute to the overall storage & no fail safe cost

- transient database and schemas are also possible

- schemas created under transient db and/or schema are by default transient.

- once created, can not be converted to other table type.

- it support clone

7. Compare

Type	Persistent	Cloning	TimeTravel	FailSafe(DR)
Permanent(Std)	Until Dropped	Yes	0 or 1	7
Permanent(Ent+)	Until Dropped	Yes	0 or 90	7
Transient	Until Dropped	Yes	0 or 1	0
Temporary	Session Level	Yes	0 or 1	0

8. timetravel can be specified for the table during creation or later via alter

9. The Fail-safe period is not configurable for any table type.

10. Transient and temporary tables have no Fail-safe period

11. External Table (ET)

- s3, azure, google cloud storage

- External tables store file-level metadata (file path, a version identifier, and partitioning information)

- External tables support external (i.e. S3, Azure, or GCS) stages only; internal (i.e. Snowflake) stages are not supported.

- Schema on read

- No DML operation is allowed

- External table can be used to perform join operation

- Views can be created onn external table

- it is bit slower than the native SF tables

- materialized views on external table improves performance (Enterprise+ editionn)

- Every ET has a column named VALUE of type VARIANT. Additional columns might be specified. All of the columns are treated as virtual columns.

  * value (a varient type) - represent single row in external file

  * METADATA$FILENAME - a metadata column

- METADATA$FILENAME is a pseudocolumn, identifies stage data file and its path

- virtual columns - it can be created using value and psedocolumn

- No referential integrity constants on external tables are enforced by Snowflake

- The default VALUE and METADATA$FILENAME columns cannot be dropped.

- Not supported **

* Clustering Key

* Cloning

* Data Sharing

* Data in XML format

* Time travel

- Recommendation for external table

* partition the table

12. Create External Table need following access

- Database - usage

- schema - usage/create stage/create external table

- stage - usage

-- Sample code to create a temp table and insert data .. all worked well

use database TIPS_SALES_PROD;

use schema public;

create temporary table t1 (f1 numeric);

desc table t1;

insert into t1 select p_size from part_expoert limit 100;

select * from t1;

drop table t1; -- this also works

-- if you run a show table, it shows the temporary tables too (refer image)

-- if you try to run following command on temporary table, you will get error

list @%t1

SQL compilation error: Stage 'TIPS_SALES_PROD.PUBLIC."%T1"' does not exist or not authorized.

drop table t2

create transient table t2 (parts_name varchar(1000)); --Table T2 successfully created.

insert into t2 select p_name from PART_EXPORT limit 1000; -- record inserted

select * from t2 limit 100;

show tables;

list @%t2; --this works

-- duplicate table name (refer images)

select * from PART_EXPORT limit 10; --this is a permanent table

create temporary table PART_EXPORT(parts_key number(38,0), parts_name varchar(55)); -- this works

create transient table PART_EXPORT(parts_key number(38,0), parts_name varchar(55)); -- this does not

drop table PART_EXPORT; -- this will drop the tmp table

show tables;

-- how to grant access to a database

GRANT USAGE, CREATE SCHEMA ON DATABASE "TIPS_DATABASE" TO ROLE "USERADMIN";

-- External Table Examples

-- additional key param is required

create or replace stage mystage url='s3://mybucket/files/'; -- aws

create or replace stage mystage url='gcs://mybucket/files'; -- gcp

create or replace stage mystage url='azure://myaccount.blob.core.windows.net/mycontainer/files'; -- azure

-- blob.core.windows.net endpoint for all supported types of Azure blob storage accounts, including Data Lake Storage Gen2.

-- create external table with stage reference

-- (Tip - create an internal stage reference and try to play with external table what happens)

-- desc external table

```
desc external table emp;
```

-- show external table

```
show external tables like 'line%' in tpch.public;
```

-- manually refresh external table

```
alter external table exttable_json refresh;
```

-- Similar to the first example, but manually refresh only a path of the metadata for an external table:

```
create or replace stage mystage
  url='<cloud_platform>://twitter_feed/logs/'
  .. ;
```

-- Create the external table

-- 'daily' path includes paths in </YYYY/MM/DD/> format

```
create or replace external table daily_tweets
  with location = @twitter_feed/daily/;
```

-- Refresh the metadata for a single day of data files by date

```
alter external table exttable_part refresh '2018/08/05/';
```

-- Add an explicit list of files to the external table metadata:

```
alter external table exttable1 add files ('path1/sales4.json.gz', 'path1/sales5.json.gz');
```

--Remove an explicit list of files from the external table metadata:

```
alter external table exttable1 remove files ('path1/sales4.json.gz', 'path1/sales5.json.gz');
```

-- drop external table

drop external table t2;

https://s3.amazonaws.com/snowflake-docs/tutorials/json/server/2.6/2016/07/15/15/json_tutorial.json


```
{
 "device_type": "server",
 "events": [
  {
   "f": 83,
   "rv": "15219.64,783.63,48674.48,84679.52,27499.78,2178.83,0.42,74900.19",
   "t": 1437560931139,
   "v": {
    "ACHZ": 42869,
    "ACV": 709489,
    "DCA": 232,
    "DCV": 62287,
    "ENJR": 2599,
    "ERRS": 205,
    "MXEC": 487,
    "TMPI": 9
   },
   "vd": 54,
   "z": 1437644222811
  },
  {
   "f": 1000083,
   "rv": "8070.52,54470.71,85331.27,9.10,70825.85,65191.82,46564.53,29422.22",
   "t": 1437036965027,
   "v": {
```

```
      "ACHZ": 6953,
      "ACV": 346795,
      "DCA": 250,
      "DCV": 46066,
      "ENJR": 9033,
      "ERRS": 615,
      "MXEC": 0,
      "TMPI": 112
    },
    "vd": 626,
    "z": 1437660796958
  }
 ],
 "version": 2.6
}
```

Data Type In Snowflake

Snowflake has its internal way of storing data, so on surface it support many style but many of them are just alias or same like int and decimal are all numbers

1. **Numeric** - Number (38,0) (rest all numbers are stored with this data type no matter what definition you give) = decimal,float,real, int, big int, etc...

2. String/Binary Data Type

= string/text/varchar

= char -> varchar(1)

= Binary = varbinary

3. Logical Data Type = Boolean (account supported post 25 Jan 2016)

4. Date/Time = Date

Time

DateTime

TimeStamp

TimeStamp_LTZ (Local Time Zone)

TimeStamp_NTZ (with no timezone)

TimeStamp_TZ (with timezone)

5. Semi Structured - Variant

Object

Array

Note : Snowflake displays FLOAT, FLOAT4, FLOAT8, REAL, DOUBLE, and DOUBLE PRECISION as FLOAT

Key Note about Numeric Data Type

- Numeric Data types are fixed point (int, bigint etc), floating point (float/double), constant/literal

- Number up to 38 with optional precision

- Precition - total number of digit allowed

- scale - totla # of digit right to the decimal point (37)

- data converted from higher to lower and back to higher, it will lose precision

- Example : convert a NUMBER(38,37) to DOUBLE (has 17 prevision), and then back to NUMBER, it lose its precision

- Precision (total number of digits) does not impact storage <***IMP***>

- SF Micropartition determine min/max precision for given column and then store <***IMP***>

- Example: col values 1,2,3,4,5 -> 1 byte uncompressed

  col values 1,2,3,100000 -> 4 byte uncompressed

- scale (number of digits following the decimal point) does have an impact on storage

- column of type NUMBER(10,5) consumes more space than NUMBER(5,0) + processing cost is higher

Key Note about Floating Point Numbers

- SF uses double-precision (64 bit) IEEE 754 floating point numbers.

- SF Supports following special values for Float

  'NaN' -> Not a number

  'inf' -> infinity

  '-inf' -> negative infinity

  ** these symbols must be single quote

- IEEE 745 vs SF when apply condition check

  if('NaN' = 'NaN') .. in SF it is true but not in IEEE

  IF('NaN' > X) ....in SF it is true but not in IEEE

Constant/Literals for Numbers

- e/E indicate exponent (At least one digit must follow the exponent marker if present.)

- Example 1.234E2 , 1.234E+2 , 15e-03

- +/- indicate postive or negative, default is +

Key Note about String & Binary Datatype

- VARCHAR holds unicode characters.

- max length is 16 MB (uncompressed) (single byte 16,777,216 and 2bypte/4byte per char - 8,388,608 to 4,194,304)

- If a length is not specified, the default is the maximum length. <***IMP***>

- char/character same as varchar and not defined it is 1 default.

- for char datatype, no padding givein like standard database.

Key Note about Binary

- Max length = 8Mb

- No notion of unicode charater, so length is always same.

- varbinary = binary

- internal representation - The BINARY data type holds a sequence of 8-bit bytes.

- HELP is stored and will display as 48454C50 (H=48 in ascii)

String Constant

- String constants in SF must always be enclosed between delimiter characters.

- SF supports using either single quotes or dollar signs to delimit string constants <***IMP***>

  Tip: in standard database, single quote is used for constant in select statement

- select 'today' as today, $$t o d a y$$ as today_with_dollar

- select $$wow \t wow$$ as today

- Two single quotes is not the same as the double quote character

- $$string with a ' character$$

- Note that the string constant cannot contain double-dollar signs. <***IMP***>

Boolean Data Type & Key Note

- BOOLEAN can have TRUE/FALSE values.

- BOOLEAN can also have an “unknown” value, which is represented by NULL

- Boolean Conversionn

  *** String**

* true/t/yes/y/on/1 = True

* false/f/no/f/off/0 = False

*** Numeric**

* 0 - zero is false

* any non zero (+/-) is true

- String/Numeric convertionnn

* to_string(True) -> true (lower case) -> 1 (for number)

* to_string(False) -> false (lower case) -> 0 (for number)

- Null is converted to NULL

Quik Review - Ternary Logic - three-valued logic (3VL)

- logic system with three truth values: TRUE, FALSE, and UNKNOWN

- Snowflake, UNKNOWN is represented by NULL

- When used in expressions (e.g. SELECT list), UNKNOWN results are returned as NULL values.

- When used as a predicate (e.g. WHERE clause), UNKNOWN results evaluate to FALSE.

Date/time key notes

- single DATE data type for storing dates (with no time elements)

- DATE supported formats for string constants (YYYY-MM-DD, DD-MON-YYYY, etc.)

- When a date-only format is used, the associated time is assumed to be midnight on that day.

- timestap = combined date + time

- DATETIME is an alias for TIMESTAMP_NTZ

- all accepted timestamps are valid inputs for dates

-TIME data type for storing times in the form of HH:MI:SS

- TIME supports an optional precision parameter for fractional seconds, e.g. TIME(3)

- Time precision can range from 0 (seconds) to 9 (nanoseconds). The default precision is 9

- All TIME values must be between 00:00:00 and 23:59:59.999999999

- TIME internally stores “wallclock” time, and all operations on TIME values are performed without taking any time zone into consideration

calendar support

-Snowflake uses the Gregorian Calendar for all dates and timestamps

Date and Time Constants

- Snowflake supports using string constants to specify fixed date, time, or timestamp values
- String constants must always be enclosed between delimiter characters
- example

  date '2010-09-14'

  time '10:03:56'

  timestamp '2009-09-15 10:59:43'

  create table t1 (d1 date);

  insert into t1 (d1) values (date '2011-10-29');

  value format can be set as alter session set date-input-format ='yyyy-MM-dd'

Interval Constants

-interval constants to add/subtract a period of time to/from a date, time, or timestamp

- Example

  INTERVAL '1 YEAR' represents 1 year.

  INTERVAL '4 years, 5 months, 3 hours' represents 4 years, 5 months, and 3 hours

  INTERVAL '1 year, 1 day' first adds/subtracts a year and then a day.

  INTERVAL '1 day, 1 year' first adds/subtracts a day and then a year.

-SQL

```
select to_date ('2019-02-28') + INTERVAL '1 day, 1 year'; -- result 2020-03-01
select to_date ('2019-02-28') + INTERVAL '1 year, 1 day'; -- result 2020-02-29
```

- INTERVAL is not a data type and can be used only with date/time

-- More example

```
select to_date('2018-04-15') + INTERVAL '1 year'; -- 2019-04-15
select to_time('04:15:29') + INTERVAL '3 hours, 18 minutes'; --07:33:29
```

```
select name, hire_date from employees where hire_date > current_date - INTERVAL '2 y, 3 month';

select to_date('2025-01-17') + INTERVAL
                    '1 y, 3 q, 4 mm, 5 w, 6 d, 7 h, 9 m, 8 s,
                     1000 ms, 445343232 us, 898498273498 ns'
                  as complex_interval2;


                  -- 2027-03-30 07:31:32.841
```

Simple Arithmetic for Dates

- TIME and TIMESTAMP values do not yet support simple arithmetic.
- select to_date('2018-04-15') + 1; -- 2018-04-16
-- select to_date('2018-04-15') - 4; -- 2018-04-11

-- setting the timezone

```
alter session set timestamp_type_mapping = timestamp_ntz;
create or replace table ts_test(ts timestamp);
desc table ts_test;

create or replace table ts_test2(ts timestamp_ltz);
desc table ts_test2;
```

IFF Behavior

The IFF function returns the following results for ternary logic. Given a BOOLEAN column `C`:

If `C` is:	`IFF(C, e1, e2)` evaluates to:
TRUE	`e1`
FALSE	`e2`
NULL	`e2`

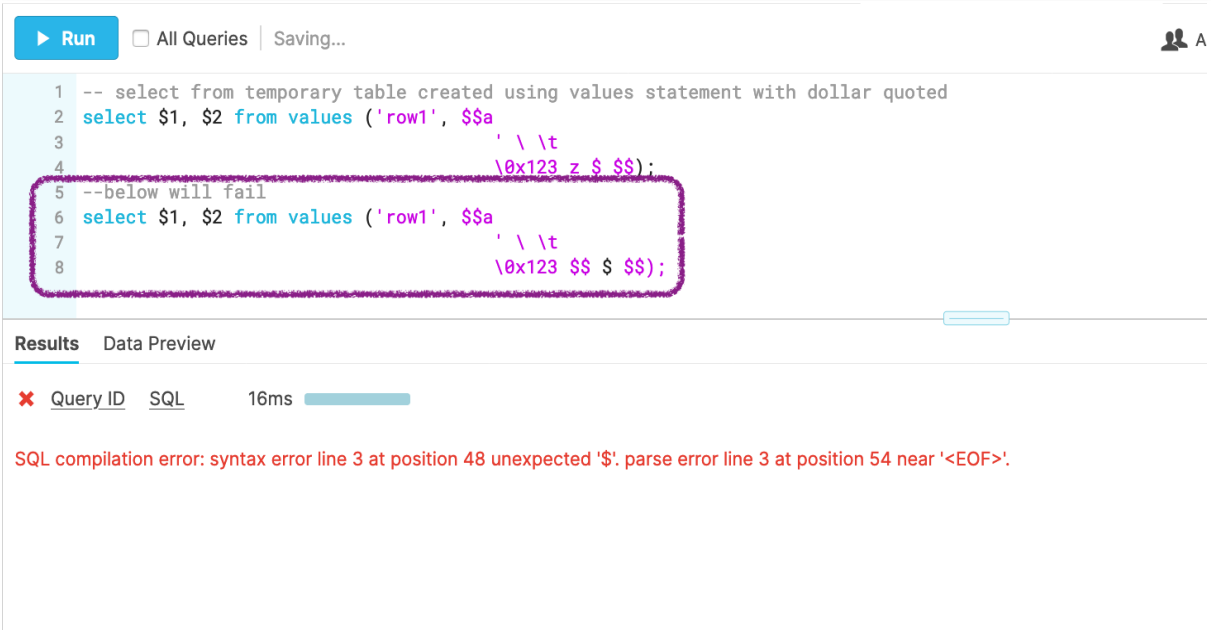
Logical Operators

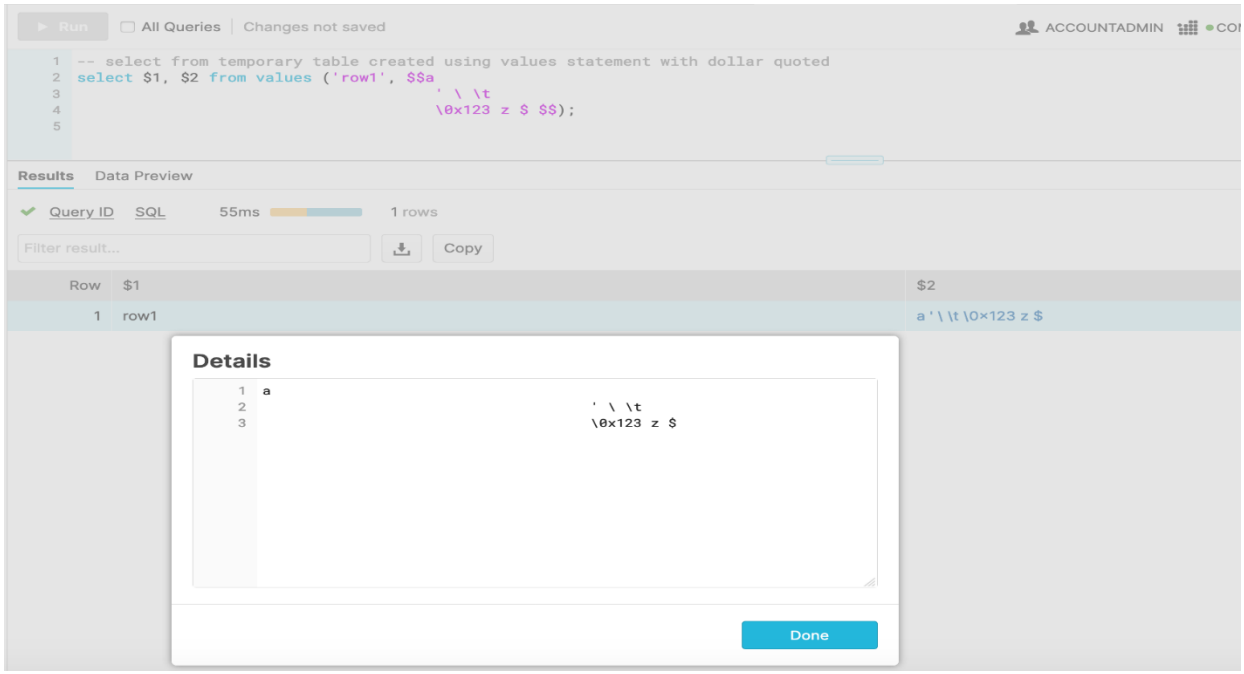
Given a BOOLEAN column `C`:

If `C` is:	`C AND NULL` evaluates to:	`C OR NULL` evaluates to:	`NOT C` evaluates to:
TRUE	NULL	TRUE	FALSE
FALSE	FALSE	NULL	TRUE
NULL	NULL	NULL	NULL

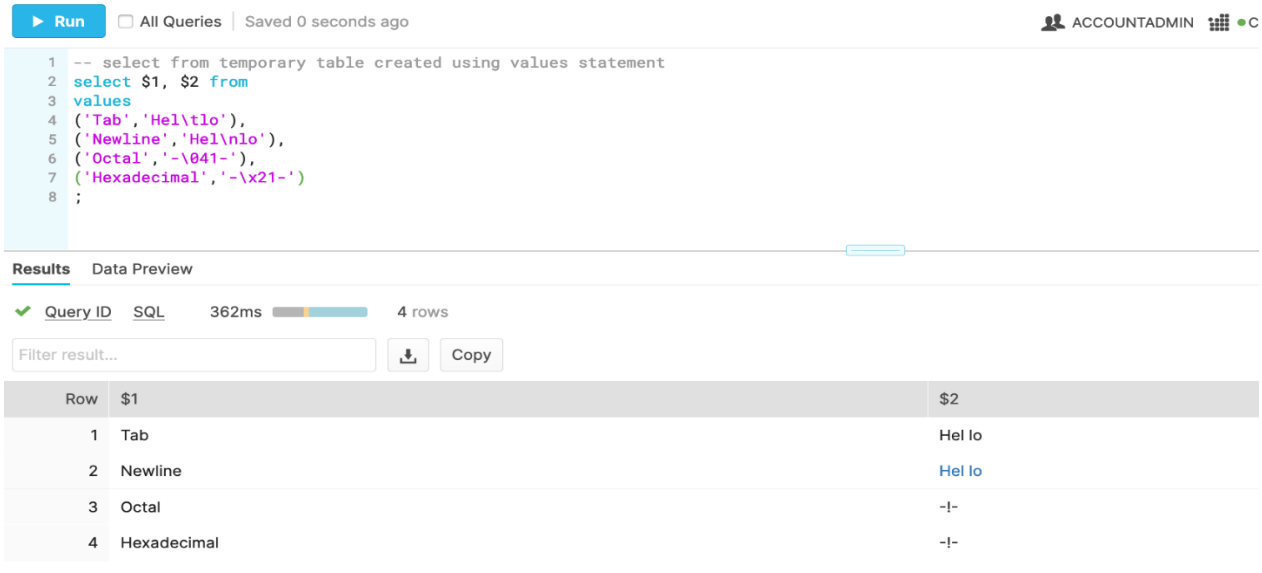
In addition:

If `C` is:	`C AND (NOT C)` evaluates to:	`C OR (NOT C)` evaluates to:	`NOT (C OR NULL)` evaluates to:
TRUE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	NULL
NULL	NULL	NULL	NULL




Run
All Queries
Changes not saved
ACCOUNTADMIN
-- select from temporary table created using values statement with dollar quoted
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 z $ $$);
Results
Data Preview
Query ID
SQL
55ms
1 rows
Filter result...
Copy
Row
$1
$2
1
row1
a ' \ \t \0×123 z $
Details
a
' \ \t
\0x123 z $
Done


Run
All Queries
Changes not saved
ACCOUNTADMIN
COMPUTE_WH
-- select from temporary table created using values statement with dollar quoted
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 z $ $$);
Results
Data Preview
Query ID
SQL
55ms
1 rows
Filter result...
Copy
Row
$1
$2
1
row1
a ' \ \t \0×123 z $


Run
All Queries
Saved 0 seconds ago
ACCOUNTADMIN
-- select from temporary table created using values statement
select $1, $2 from
values
('Tab','Hel\tlo'),
('Newline','Hel\nlo'),
('Octal','-\041-'),
('Hexadecimal','-\x21-')
;
Results
Data Preview
Query ID
SQL
362ms
4 rows
Filter result...
Copy
Row
$1
$2
1
Tab
Hel lo
2
Newline
Hel lo
3
Octal
-!-
4
Hexadecimal
-!-

Run | All Queries | Saved 0 seconds ago

```
1  select 'today' as today
```

Results | Data Preview

Query ID | SQL | 51ms | 1 rows

Filter result... | Copy

Row	TODAY
1	today

Run | All Queries | Saved 0 seconds ago

```
1  -- create table with binary
2  create or replace table test_binary(b binary,
3                                      b100 binary(100),
4                                      vb varbinary
5                                      );
6  desc table test_binary;
7  --insert data
8  insert into test_binary values ('HELP', 'help', 'Help')
9
10
```

Results | Data Preview

Query ID | SQL | 1.21s

String 'HELP' is not a legal hex-encoded string

```
-- create table with binary
create or replace table test_binary(b binary,
                                    b100 binary(100),
                                    vb varbinary
                                    );
desc table test_binary;
--insert data
insert into test_binary values ('HELP', 'help', 'Help')
```

Results | Data Preview

Query ID | SQL | 47ms | 3 rows

Filter result... | Copy

Row	name	type	kind	null?	default	primary key
1	B	BINARY(8388608)	COLUMN	Y	NULL	N
2	B100	BINARY(100)	COLUMN	Y	NULL	N
3	VB	BINARY(8388608)	COLUMN	Y	NULL	N

```
-- check the padding in character data type
create or replace table my_char_tab(
c char,
c1 char(1) ,
c10 char(10)
);
--insert values
insert into my_char_tab values ('a','b','c'),('x','y','zzzz')
--check length
select len(c), len(c1), len(c10) from my_char_tab
```

Results | Data Preview | Open History

Query ID | SQL | 201ms | 2 rows

Filter result... | Copy | Columns

Row	LEN(C)	LEN(C1)	LEN(C10)
1	1	1	1
2	1	1	4

No padding done as it with other databases where char has padding for rest of the space

```
-- check the padding in character data type
create or replace table my_char_tab(
c char,
c1 char(1) ,
c10 char(10)
);
--insert values
insert into my_char_tab values ('a','b','c'),('xx','y','zzzz')
```

Results | Data Preview

✖ Query ID | SQL | 518ms

String 'xx' is too long and would be truncated

```sql
-- constant example
create table my_constant(my_number number(38,30))
--insert data using constant/literals
insert into my_constant
values
(15),
(+1.34),
(0.2),
(15e-03),
(1.234E2),
(1.234E+2),
(-1)
select * from my_constant
```

Important thing to notice here that 30 zeros appears after decimal due to definition

Results | Data Preview — Open History

Query ID | SQL | 359ms | 7 rows

Filter result... | Copy | Columns

Row	MY_NUMBER
1	15.000000000000000000000000000000
2	1.340000000000000000000000000000
3	0.200000000000000000000000000000
4	0.015000000000000000000000000000
5	123.400000000000000000000000000000
6	123.400000000000000000000000000000
7	-1.000000000000000000000000000000

```sql
create or replace table test_boolean(
   b boolean,
   n number,
   s string);
   --insert some sample data
   insert into test_boolean
   values (true, 1, 'yes'),
          (false, 0, 'no'),
          (true, -1, 'on'),
          (null, null, null);

   -- casting nuber and string to boolean
   select b col_boolean, n col_num, s col_str, to_boolean(n) numb_to_boolean, to_boolean(s) str_to_boolean from test_boolean;
```

esults | Data Preview

Query ID | SQL | 1s | 4 rows

Filter result... | Copy

Row	COL_BOOLEAN	COL_NUM	COL_STR	NUMB_TO_BOOLEAN	STR_TO_BOOLEAN
1	TRUE	1	yes	TRUE	TRUE
2	FALSE	0	no	FALSE	FALSE
3	TRUE	-1	on	TRUE	TRUE
4	NULL	NULL	NULL	NULL	NULL

Run | All Queries | Saved 4 seconds ago — ACCOUNTADMIN | COMPUTE_WH (XS) | USDA_NUTRIENT_... | PUBLIC

```sql
create or replace table test_binary(b binary,
                                    b100 binary(100),
                                    vb varbinary
                                    );

desc table test_binary;
```

Results | Data Preview — Open His

Query ID | SQL | 146ms | 3 rows

Filter result... | Copy | Columns

Row	name	type	kind	null?	default	primary key	unique key	check	expression	comment
1	B	BINARY(8388608)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
2	B100	BINARY(100)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
3	VB	BINARY(8388608)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL

```
create or replace table test_text(v varchar,
                                  v50 varchar(50),
                                  c char,
                                  c10 char(10),
                                  s string,
                                  s20 string(20),
                                  t text,
                                  t30 text(30)
                                  );

desc table test_text;
```

sults Data Preview — Open History

Query ID SQL 53ms 8 rows

ter result... Copy Columns

Row	name	type	kind	null?	default	primary key	unique key	check	expression	comment
1	V	VARCHAR(16777216)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
2	V50	VARCHAR(50)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
3	C	VARCHAR(1)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
4	C10	VARCHAR(10)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
5	S	VARCHAR(16777216)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
6	S20	VARCHAR(20)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
7	T	VARCHAR(16777216)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
8	T30	VARCHAR(30)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL

Run All Queries Saved 3 seconds ago — ACCOUNTADMIN COMPUTE_WH (XS) USDA_NUTRIENT_... PUBLIC

```
create or replace table test_float(d double,
                                   f float,
                                   dp double precision,
                                   r real,
                                  d_num number(38,8)
                                   );

                                   desc table test_float;
                                   drop table test_float
```

Results Data Preview — Open H

Query ID SQL 59ms 5 rows

Filter result... Copy Columns

Row	name	type	kind	null?	default	primary key	unique key	check	expression	comment
1	D	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
2	F	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
3	DP	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
4	R	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL
5	D_NUM	NUMBER(38,8)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL

Examples of Floating Point Data Types in a Table

```
create or replace table test_float(d double,
                                   f float,
                                   dp double precision,
                                   r real
                                   );

desc table test_float;
```

```
+---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------+
| name    | type         | kind   | null? | default | primary key | unique key | check | expression | comment |
|---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------|
| D       | FLOAT        | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| F       | FLOAT        | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| DP      | FLOAT        | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| R       | FLOAT        | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
+---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------+
```

Examples of Fixed-point Data Types in a Table

```
create or replace table test_fixed(num number,
                                   num10 number(10,1),
                                   dec decimal(20,2),
                                   numeric numeric(30,3),
                                   int int,
                                   integer integer
                                   );

desc table test_fixed;
```

```
+---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------+
| name    | type         | kind   | null? | default | primary key | unique key | check | expression | comment |
|---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------|
| NUM     | NUMBER(38,0) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| NUM10   | NUMBER(10,1) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| DEC     | NUMBER(20,2) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| NUMERIC | NUMBER(30,3) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| INT     | NUMBER(38,0) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
| INTEGER | NUMBER(38,0) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    |
+---------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------+
```

Supported Date and Time Parts for Intervals

The INTERVAL keyword supports the following date and time parts as arguments (case-insensitive):

Date or Time Part	Abbreviations / Variations
year	y , yy , yyy , yyyy , yr , years , yrs
quarter	q , qtr , qtrs , quarters
month	mm , mon , mons , months
week	w , wk , weekofyear , woy , wy , weeks
day	d , dd , days , dayofmonth
hour	h , hh , hr , hours , hrs
minute	m , mi , min , minutes , mins
second	s , sec , seconds , secs
millisecond	ms , msec , milliseconds
microsecond	us , usec , microseconds
nanosecond	ns , nsec , nanosec , nsecond , nanoseconds , nanosecs , nseconds

```
6
7  create or replace table ts_test2(ts timestamp_ltz);
8
9  desc table ts_test2;
```

Results Data Preview

Query ID SQL 57ms 1 rows

Filter result... Copy

Row	name	type	kind	null?	default
1	TS	TIMESTAMP_LTZ(9)	COLUMN	Y	NULL

```
7 create or replace table ts_test2(ts timestamp_ltz);
8
9 desc table ts_test2;
```

Results | Data Preview

Query ID | SQL | 57ms | 1 rows

Filter result... | Copy

Row	name	type	kind	null?	default	primary key
1	TS	TIMESTAMP_LT...	COLUMN	Y	NULL	N

Run | All Queries | Saved 19 seconds ago | ACCOUNTADMIN

```
1 alter session set timestamp_type_mapping = timestamp_ntz;
2
3 create or replace table ts_test(ts timestamp);
4
5 desc table ts_test;
```

Results | Data Preview

Query ID | SQL | 68ms | 1 rows

Filter result... | Copy

Row	name	type	kind	null?	default	primary key
1	TS	TIMESTAMP_NTZ(9)	COLUMN	Y	NULL	N

Variant Data Type

-- Variant/Array/Object Data Type

Semi-structure data type: used to represent arbitrary data structures

Can be used to import/operate on semi-structured data

JSON

Avro

ORC

Parquet

XML

Snowflake stores these types internally in an efficient compressed columnar binary representation

Variant

----

universal type, can store values of any other type, including OBJECT and ARRAY

maximum size of 16 MB compressed

A value of any data type can be implicitly cast to a VARIANT value, subject to size restrictions

example : (var:json_path >= 56) is cast to (var:json_path >= 56::VARIANT) for compare

VARIANT value can be missing (contain SQL NULL), which is different from a VARIANT null value

** VARIANT columns in a relational table are stored as separate physical columns

Object

Used to represent collections of key-value pairs,

key is a non-empty string, and the value is a value of VARIANT type

Snowflake does not currently support explicitly-typed objects.

Array

Used to represent dense or sparse arrays of arbitrary size

Unsupported Data Type

LOB (Large Object) - BINARY can be used instead

CLOB - VARCHAR can be used instead;

ENUM

User-defined data types

Querying Semi-structured Data

Snowflake supports SQL queries that access semi-structured data

JSON, Avro, ORC, and Parquet data (but not to xml)

Traverse Pattern Insert a colon : between the VARIANT column name and any first-level element

<column>:<level1_element>.

Query output is enclosed in double quotes because the query output is VARIANT, not VARCHAR

Operators : and subsequent . and [] always return VARIANT values containing strings

There are two ways to access elements in a JSON object:

Dot Notation

Bracket Notation

Regardless of which notation you use, the column name is case-insensitive but element names are case-sensitive

src:salesperson.name

SRC:salesperson.name

SRC:Salesperson.Name

Dot notation

```
select src:salesperson.name from car_sales;
```

Bracker notation -- Enclose element names in single quotes

```
select src['salesperson']['name'] from car_sales;
```

Repeating elements

```
select src:vehicle[0] from car_sales;

select src:vehicle[0].price from car_sales;
```

Explicitly Casting

```
select src:salesperson.id::string from car_sales;
```

FLATTEN Function to Parse Arrays

FLATTEN is a table function that produces a lateral view of a VARIANT, OBJECT, or ARRAY

Example:

```
select
   value:name::string as "Customer Name",
   value:address::string as "Address"
from
   car_sales
, lateral flatten(input => src:customer);
```

Json Data Load

#Sample JSON Data

[{"time":"6:38:51.000 PM","customer":"Talya McCambrois","action":"power off","device":"Footbot Air Quality Monitor"},

{"time":"8:13:16.000 PM","customer":"Willi Jenkerson","action":"power off","device":"GreenIQ Controller"},

{"time":"2:21:46.000 AM","customer":"Kippy Roux","action":"power off","device":"Amazon Echo"}]

```
CREATE DATABASE "TIPS_DATABASE"


CREATE FILE FORMAT "TIPS_DATABASE"."LAB".IOT_JSON

TYPE = 'JSON'

COMPRESSION = 'AUTO'

ENABLE_OCTAL = FALSE

ALLOW_DUPLICATE = TRUE

STRIP_OUTER_ARRAY = FALSE

STRIP_NULL_VALUES = TRUE

IGNORE_UTF8_ERRORS = FALSE

COMMENT = 'JSON File Format';
```

-- Step 4: Creating an internal stage to load data

```
CREATE STAGE "TIPS_DATABASE"."LAB".IOT_LAB_STAGE

COMMENT = 'This is iot data loading stage';
```

-- Step 4: Creating a table

```
CREATE TABLE "TIPS_DATABASE"."LAB"."DEVICE_DATA" (

    "EVENT_TIME" TIMESTAMP NOT NULL,
```

```
   "CUSTOMER_NAME" VARCHAR (100) NOT NULL,
   "DEVICE_ACTION" VARCHAR (50) NOT NULL,
   "DEVICE_NAME" VARCHAR (200) NOT NULL)
COMMENT = 'This is customer iot device data';
```

PUT file://<file_path>/1.json @DEVICE_DATA/ui1589027218487

```
COPY INTO "TIPS_DATABASE"."LAB"."DEVICE_DATA"
   FROM @/ui1589027218487
   FILE_FORMAT = '"TIPS_DATABASE"."LAB"."IOT_JSON"'
   ON_ERROR = 'CONTINUE'
   PURGE = TRUE;
```

#since file has issue.. it threw following error

Unable to copy files into table.

SQL compilation error: JSON/XML/AVRO file format can produce

one and only one column of type variant or object or array.

Use CSV file format if you want to load more than one column.

```
CREATE TABLE "TIPS_DATABASE"."LAB"."DEVICE_DATA_VARIANT" ("DEVICE_EVENT" VARIANT NOT NULL) COMMENT = 'This is single column data';
```

-- Data loaded into single row

-- approach 2 to have in mutlipel rows

```
CREATE TABLE "TIPS_DATABASE"."LAB"."DEVICE_DATA_VARIANT_2" ("DEVICE_EVENT" VARIANT NOT NULL) COMMENT = 'Trying to load on individual rows';
```

```
CREATE FILE FORMAT "TIPS_DATABASE"."LAB".Device_Event_Format TYPE = 'JSON' COMPRESSION = 'AUTO' ENABLE_OCTAL = FALSE ALLOW_DUPLICATE = TRUE STRIP_OUTER_ARRAY = TRUE STRIP_NULL_VALUES = TRUE IGNORE_UTF8_ERRORS = TRUE COMMENT = 'Each element as single row';
```

PUT file://<file_path>/1.json @DEVICE_DATA_VARIANT_2/ui1589027881962

COPY INTO "TIPS_DATABASE"."LAB"."DEVICE_DATA_VARIANT_2" FROM @/ui1589027881962 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."DEVICE_EVENT_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- table to load ORC file via structured schema

CREATE TABLE "TIPS_DATABASE"."LAB"."ORC_DEVICE_DATA" ("EVENT_TIME" TIMESTAMP NOT NULL, "CUSTOMER_NAME" VARCHAR (100) NOT NULL, "EVENT_ACTION" VARCHAR (50) NOT NULL, "DEVICE_NAME" VARCHAR (200) NOT NULL) COMMENT = 'Storing Data from ORC File';

-- create file format to load data to table via put command

CREATE FILE FORMAT "TIPS_DATABASE"."LAB".ORC_DATA_FORMAT TYPE = 'ORC' COMMENT = 'This is ORC data format file';

-- data loading command via internal unmammed stage

PUT file://<file_path>/part-00000-921b6fdc-41bb-4b06-b5d8-a8b760cb1558-c000.snappy.orc @ORC_DEVICE_DATA/ui1589034165777

COPY INTO "TIPS_DATABASE"."LAB"."ORC_DEVICE_DATA" FROM @/ui1589034165777 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."DEVICE_EVENT_FORMAT"' ON_ERROR = 'ABORT_STATEMENT' PURGE = TRUE;

-- looks it also gave an error like other data set

Unable to copy files into table.

SQL compilation error: JSON/XML/AVRO file format can produce one

and only one column of type variant or object or array.

Use CSV file format if you want to load more than one column.

-- Now if you have lot of large data set then it will be an expensive operation

CREATE TABLE "TIPS_DATABASE"."LAB"."ORC_DEVICE_DATA_VARIANT" ("DEVICE_DATA" VARIANT NOT NULL) COMMENT = 'This table will have only one column';

PUT file://<file_path>/part-00000-921b6fdc-41bb-4b06-b5d8-a8b760cb1558-c000.snappy.orc @ORC_DEVICE_DATA_VARIANT/ui1589034420521

COPY INTO "TIPS_DATABASE"."LAB"."ORC_DEVICE_DATA_VARIANT" FROM @/ui1589034420521 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."ORC_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- Now parquet type

CREATE FILE FORMAT "TIPS_DATABASE"."LAB".PARQUET_DATA_FORMAT TYPE = 'PARQUET' COMPRESSION = 'SNAPPY' BINARY_AS_TEXT = TRUE COMMENT = 'File Format for Parquet Data';

-- load command

PUT file://<file_path>/part-00000-d5a70322-eb9d-4e02-a85f-e678c1029748-c000.snappy.parquet

@ORC_DEVICE_DATA/ui1589034857768

COPY INTO "TIPS_DATABASE"."LAB"."ORC_DEVICE_DATA"

FROM @/ui1589034857768

FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"'

ON_ERROR = 'CONTINUE'

PURGE = TRUE;

it also ended in errorUnable to copy files into table.

SQL compilation error: JSON/XML/AVRO file format can produce one and only one column of type variant or object or array. Use CSV file format if you want to load more than one column.

-- Now defineing a new table

CREATE TABLE "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_VARIANT" ("DEVICE_EVENT" VARIANT) COMMENT = 'The single column data loaded via Variant';

-- different options

PUT file://<file_path>/part-00000-d5a70322-eb9d-4e02-a85f-e678c1029748-c000.snappy.parquet @PARQUET_DEVIDE_DATA_VARIANT/ui1589035032039

--Do not load any data in the file

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_VARIANT" FROM @/ui1589035032039 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'SKIP_FILE' PURGE = TRUE;

--Stop loading, rollback and return the error

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_VARIANT" FROM @/ui1589035032039 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'ABORT_STATEMENT' PURGE = TRUE;

--Do not load any data in the file if the error count exceeds:

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_VARIANT" FROM @/ui1589035032039 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'SKIP_FILE_100' PURGE = TRUE;

--Continue loading valid data from the file

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_VARIANT" FROM @/ui1589035032039 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- Now trying to understand the object type

CREATE TABLE "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_OBJECT" ("EVENT_RECORD" OBJECT NOT NULL) COMMENT = 'This table has one field with object as field type';

-- loading data via put and copy command

PUT file://<file_path>/part-00000-d5a70322-eb9d-4e02-a85f-e678c1029748-c000.snappy.parquet @PARQUET_DEVIDE_DATA_OBJECT/ui1589035396167

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVIDE_DATA_OBJECT" FROM @/ui1589035396167 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- ended up with error -- not sure why

Unable to copy files into table.

SQL execution internal error: Processing aborted due to error 300010:3163851172; incident 9323353.

-- Now the final one is to use the array

CREATE TABLE "TIPS_DATABASE"."LAB"."PARQUET_DEVICE_DATA_ARRAY" ("DEVICE_EVENT" ARRAY NOT NULL) COMMENT = 'This is an experiment with Parquet with array';

PUT file://<file_path>/part-00000-d5a70322-eb9d-4e02-a85f-e678c1029748-c000.snappy.parquet @PARQUET_DEVICE_DATA_ARRAY/ui1589035563444

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVICE_DATA_ARRAY" FROM @/ui1589035563444 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- end up with error -- could be due to the role issue as format is created by different role

Unable to copy files into table.

SQL execution internal error: Processing aborted due to error 300010:3163851172; incident 5140068.

-- Now following is given..lets see what happens

GRANT UPDATE, INSERT, DELETE, REBUILD, REFERENCES, SELECT, TRUNCATE

ON TABLE "TIPS_DATABASE"."LAB"."PARQUET_DEVICE_DATA_ARRAY"

TO ROLE "SYSADMIN" WITH GRANT OPTION;

-- Now role is changed

PUT file://<file_path>/tips.snappy.parquet @PARQUET_DEVICE_DATA_ARRAY/ui1589035944814

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_DEVICE_DATA_ARRAY" FROM @/ui1589035944814 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- got the error like this

There was an error while trying to stage "tips.snappy.parquet".

Reason: SQL access control error: Insufficient privileges to operate on table stage 'PARQUET_DEVICE_DATA_ARRAY'

-- looks the internal stage has issues ???? looks a big issue -- looks the int stage has access issue

-- so createa a table again with sysadmin

CREATE TABLE "TIPS_DATABASE"."LAB"."PARQUET_TYPE_ARRAY" ("DEVICE_EVENT" ARRAY NOT NULL) COMMENT = 'Parquet data loaded in array data type';

-- step-2

PUT file://<file_path>/tips.snappy.parquet @PARQUET_TYPE_ARRAY/ui1589036262859

COPY INTO "TIPS_DATABASE"."LAB"."PARQUET_TYPE_ARRAY" FROM @/ui1589036262859 FILE_FORMAT = '"TIPS_DATABASE"."LAB"."PARQUET_DATA_FORMAT"' ON_ERROR = 'CONTINUE' PURGE = TRUE;

-- step-3 looks error again

Unable to copy files into table.

SQL execution internal error: Processing aborted due to error 300010:3163851172; incident 5508373.

-- @TOD - understant the object and array data type and why it is not not working with parquet.

Data Unloading Export

-- Data Unloading & Export Function

We all know that Snowflake is cloud native warehouse system, data unloading and export function cost.

Hence it is very important to understand the data unloading practices and follow the guidelines.

There are various ways data can be exported effectively (like cloud resion, buckets, local box etc) and following topics will help us to explore them in detail.

Points to remember

1. Snowflake supports bulk export (i.e. unload) of data from a database table into flat, delimited text files

2. Use 'COPY INTO <location>' command (similar to load) the data from the Snowflake database table into one or more files in a Snowflake or external stage.

   (Tip: it is alwasy stages?)

3. Use the get command to download the file.

   (Tip: so it is 2 stage process.. like a actual warehouse.. get the item at stage area and then get it in your truck)

4. Select query can also be used instead of 'COPY INTO <location>'

   (Tip: look for code syntax)

5. Select query support full syntax including JSON style to download.

6. 'COPY INTO <location>' command provides option to specify single file or multiple (default SINGLE = FALSE)

7. Snowflake provides each file a unique name

   (Tip: try this out to reinformce the understanding and file name patters)

8. Snowflake prefixes the generated filenames with "data_".

9. Snowflake appends a suffix that ensures each file name is unique across parallel execution threads; e.g. data_stats_0_1_0.

10. MAX_FILE_SIZE copy option to specify each file

    (Tip: mb or gb or what?)

11. Compression/Encryption is decideds if stage is external or internal.

12. Location of stages - Internal (local), AWS s3, GCP bucket, Azure file

13. File formats - delited files (CSV/TSV), JSON, Parquet

(Tip: not supported ORC, xml)

14. File encoding - UTF-8

15. Compression - gzip (default)/bzip2/Brotil/ZStandard

16. File Encryption - for internal stages - 128-bit (256-bit need configuration)

17. File Encryption - for external - user supplied key (possible only if it is provided)

Super Important Notes

1. An empty string is a string with zero length or no characters, whereas NULL values represent an absence of data

2. For CSV (delimited files)

2.1 An empty string - ,'',, to indicate that the string contains zero characters

2.2 a NULL value is typically represented by two successive delimiters (e.g. ,,)

(Tip: get your hands on to reinforce theory and also try with TSV)

3. FIELD_OPTIONALLY_ENCLOSED_BY = 'character' | NONE - use this option to define behaviour (PREFERRED ONE****)

4. EMPTY_FIELD_AS_NULL = TRUE|FALSE option set to FALSE

5. NULL_IF = ('<string>',[,'<string>']) : When unloading data from tables: Snowflake converts SQL NULL values to the first value in the list.

(Tip: see all the option in play .. refer images)

6. ESCAPE_UNENCLOSED_FIELD = \\N (default)

-- Setting the param for ui session with

-- worksheet to work with specific role/warehouse/db/schema

```
use role sysadmin;

use warehouse compute_wh;

use database TIPS_SALES_PROD;

use schema public;
```

-- populate data to a table from the example table

```
CREATE TABLE part_export  LIKE snowflake_sample_data.tpch_sf1.part
INSERT INTO part_export select * from snowflake_sample_data.tpch_sf1.part
```

-- copying data to stage in csv format

```
copy into @export_stg/default_csv from part_export
```

-- all records loaded to stage and md5 check is also generated

-- the default_csv will be the prefix _0_0_0.csv.gz

-- Note: if you don't give a name and if stage is having data followign error will come

-- <error> Files already existing at the unload destination: @export_stg. Use overwrite option to force unloading.

-- use remove function as alterntively

```
REMOVE @export_stg/export_stg/default_csv_0_0_0.csv.gz
```

-- Creating a table

```
create or replace table tips_export(
id number(8) not null,
first_name varchar(255) default null,
last_name varchar(255) default null,
city varchar(255),
state varchar(255)
);
```

-- Populate the table with data

```
insert into tips_export (id,first_name,last_name,city,state)
 values
(1,'Goutam','Shetty','Salt Lake City','UT'),
(2,'Rue','Shane','Birmingham','AL'),
```

(3,'John','Heck','Columbus','GA'),

(3,'John','Heck','Columbus','GA'),

(3,'Pick','Me','New York','NY'),

(3,'Drop','You','Columbus','GA'),

(3,'Hell','Nick','Columbus','GA'),

(3,'Sim','Jack','Columbus','GA'),

(3,'Jim','Raw','Columbus','GA'),

(3,'Tium','Keo','New York','NY');

-- create a stage

CREATE STAGE export_stg COMMENT = 'my_simple_stage';

-- initiate a copy command and load data in json using selecd query

copy into @export_stg

from (select object_construct('id', id, 'first_name', first_name, 'last_name', last_name, 'city', city, 'state', state) from tips_export)

file_format = (type = json);

-- Data unload in parquet format

copy into @export_stg/myfile.parquet

from (select id, first_name, last_name, city,state from tips_export)

file_format=(type='parquet')

header = true;

-- export multiple & compression

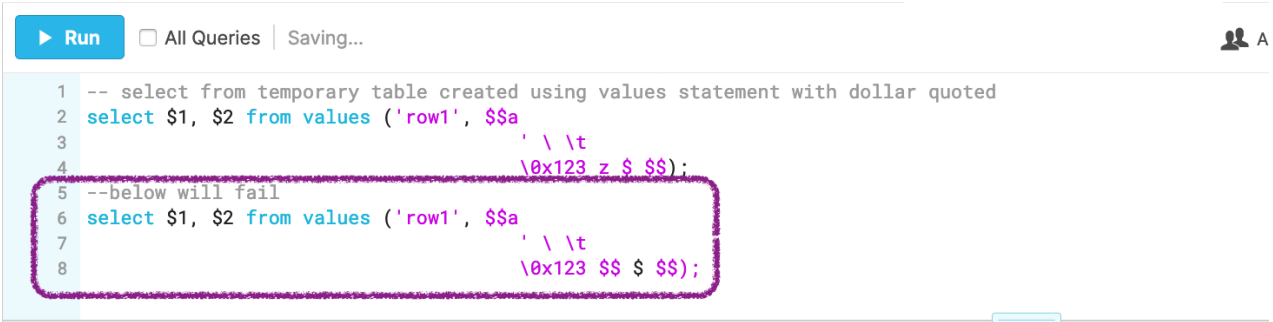
REMOVE @export_stg/file_size_0_0_0.csv.gz

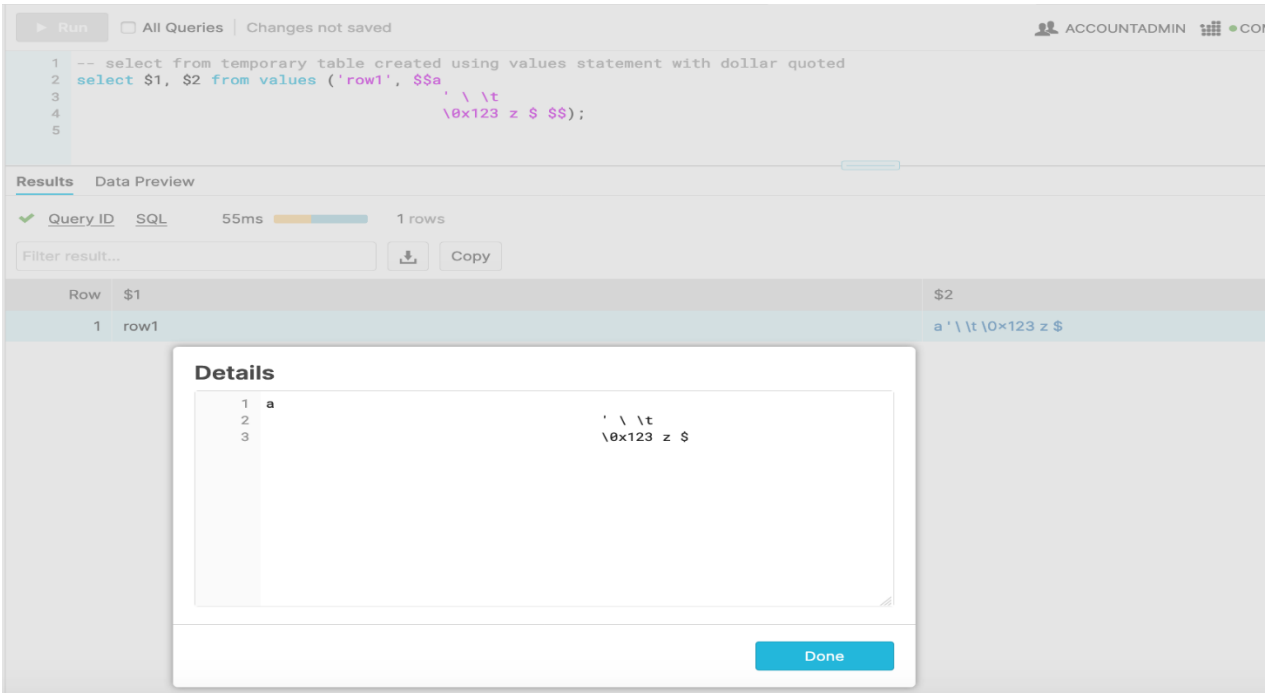
REMOVE @export_stg/file_size_0_0_1.csv.gz

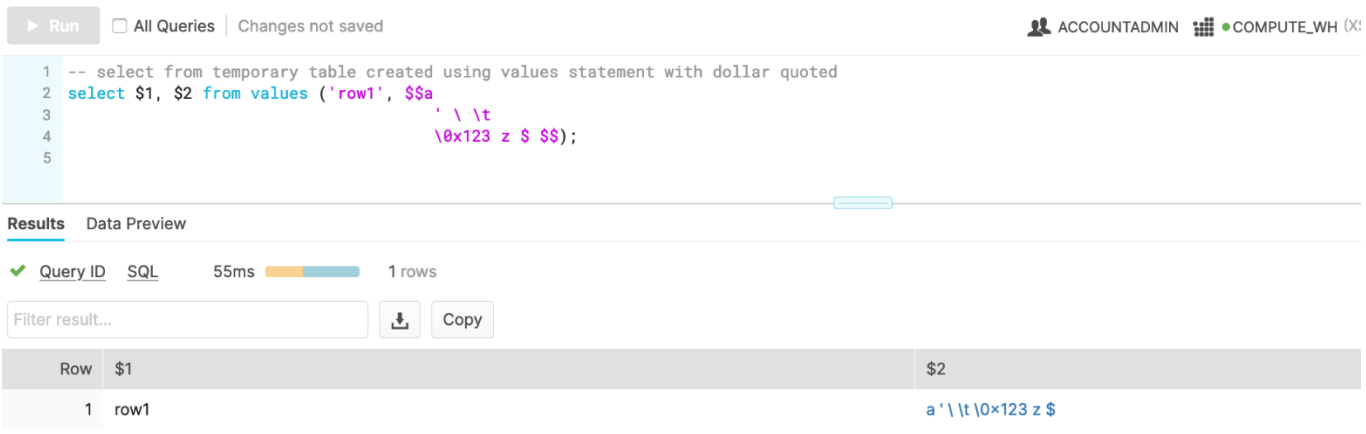
copy into @export_stg/file_size from part_export

file_format = (type=csv compression='gzip')

max_file_size=102400;


Run
All Queries
Saving...
A
-- select from temporary table created using values statement with dollar quoted
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 z $ $$);
--below will fail
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 $$ $ $$);
Results
Data Preview
Query ID
SQL
16ms
SQL compilation error: syntax error line 3 at position 48 unexpected '$'. parse error line 3 at position 54 near '<EOF>'.


Run
All Queries
Changes not saved
ACCOUNTADMIN
-- select from temporary table created using values statement with dollar quoted
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 z $ $$);
Results
Data Preview
Query ID
SQL
55ms
1 rows
Filter result...
Copy
Row
$1
$2
1
row1
a ' \ \t \0×123 z $
Details
a
' \ \t
\0x123 z $
Done


Run
All Queries
Changes not saved
ACCOUNTADMIN
COMPUTE_WH
-- select from temporary table created using values statement with dollar quoted
select $1, $2 from values ('row1', $$a
' \ \t
\0x123 z $ $$);
Results
Data Preview
Query ID
SQL
55ms
1 rows
Filter result...
Copy
Row
$1
$2
1
row1
a ' \ \t \0×123 z $

Run | All Queries | Saved 0 seconds ago | ACCOUNTADMIN | C

```
-- select from temporary table created using values statement
select $1, $2 from
values
('Tab','Hel\tlo'),
('Newline','Hel\nlo'),
('Octal','-\041-'),
('Hexadecimal','-\x21-')
;
```

Results | Data Preview

Query ID | SQL | 362ms | 4 rows

Filter result... | Copy

Row	$1	$2
1	Tab	Hel lo
2	Newline	Hel lo
3	Octal	-!-
4	Hexadecimal	-!-

Run | All Queries | Saving... | ACCOUNTADMIN | COMPUTE_WH (XS) | USDA_NUTRIENT_... | PUBLIC

```
-- create table with binary
create or replace table test_binary(b binary,
                                    b100 binary(100),
                                    vb varbinary
                                    );
desc table test_binary;
--insert data
insert into test_binary
select to_binary('ab'), to_binary('AB'), to_binary('Ab')

select * from test_binary
-- insert into demo_binary (b) select to_binary(hex_encode('HELP'), 'HEX');
```

Results | Data Preview | Open History

Query ID | SQL | 176ms | 1 rows

Filter result... | Copy | Columns

Row	B	B100	VB
1	AB	AB	AB

Run | All Queries | Saved 0 seconds ago

```
-- create table with binary
create or replace table test_binary(b binary,
                                    b100 binary(100),
                                    vb varbinary
                                    );
desc table test_binary;
--insert data
insert into test_binary values ('HELP', 'help', 'Help')

```

Results | Data Preview

Query ID | SQL | 1.21s

String 'HELP' is not a legal hex-encoded string

```
-- create table with binary
create or replace table test_binary(b binary,
                                    b100 binary(100),
                                    vb varbinary
                                    );
desc table test_binary;
--insert data
insert into test_binary values ('HELP', 'help', 'Help')
```

Results | Data Preview

Query ID | SQL | 47ms | 3 rows

Filter result... | Copy

Row	name	type	kind	null?	default	primary key
1	B	BINARY(8388608)	COLUMN	Y	NULL	N
2	B100	BINARY(100)	COLUMN	Y	NULL	N
3	VB	BINARY(8388608)	COLUMN	Y	NULL	N

```
copy into @export_stg/myfile.parquet
from (select id, first_name, last_name, city,state  from tips_export)
file_format=(type='parquet')
header = true;
```

esults | Data Preview

Query ID | SQL | 320ms | 1 rows

Filter result... | Copy

Row	rows_unloaded	input_bytes
1	10	1517

```
copy into @export_stg
from (select object_construct('id', id, 'first_name', first_name, 'last_name', last_name, 'city', city, 'state', state) from tips_export)
file_format = (type = json);

list @export_stg
```

Total Duration	108ms
Compilation	27ms
Execution	81ms
Compiling SQL	27ms
Executing (serial)	48ms
Receiving request from client	33ms

esults | Oper

Query ID | SQL | 108ms | 1 rows

Filter result... | Copy | Columns

Row	name	size	md5	last_modified
1	export_stg/data_0_0_0.json.gz	224	01e1c7f53ff04157d75de142a4c94901	Thu, 14 May 2020 08:05:09 GMT

```
copy into @export_stg
from (select object_construct('id', id, 'first_name', first_name, 'last_name', last_name, 'city', city, 'state', state) from tips_export)
file_format = (type = json);

list @export_stg
```

esults | Data Preview | Open H

Query ID | SQL | 108ms | 1 rows

Filter result... | Copy | Columns

Row	name	size	md5	last_modified
1	export_stg/data_0_0_0.json.gz	224	01e1c7f53ff04157d75de142a4c94901	Thu, 14 May 2020 08:05:09 GMT

Snowflake Security Overview

VPC - Virtual Private Cloud (Virtual Private Snowflake VPS)

Physical Security

24hr arm guards in data center

video survilience data center

no access to any unauthorized personality to data center

not Snoflake personal nor Snowflake customer have access to these data centers

Data Redundancy

Provided by cloud provider

Network Access

Network Policy helps to control the SF access

IP Whitelisting - policy can be created to allow or disallow IPs

private link - private tunnel between customer and cloud provider (ESD or VPS customer)

Account Access & Authentication

MFA - Multi Factor Authentication can be implemented to increase Security

MFA is provided by duo Security sevices

Once MFA is installed, Duo app to be installed by user

Each user must enable MFA by themself.

All user with account admin role should have MFA enabled.

SSO (SAML 2.0) allows user to access via federated services (IDP Identity provider)

As long as IPD session is active, they can access SF

SSO/IDP is available enterprise edition & +

Object Security

All the objects (warehouse/db/schema/table etc) can be controlled by DAC/RBAC

DAC- Discretionary Access control

RBAC - Role Based Access control

SF implements hybrid model of DAC & RBAC

DAC handle the ownership, each object has an owner and owner has full access to the object.

RBAC - Handle all other access except ownership like object priviledge and role access

Object priviledge assign to role which are intern assign to users.

Data Security

All data is encrypted uses AES-256 strong Encryption

All files stored in stage area is automatically Encryption using AES-128 or AES-256

Special edition of SF allows periodic re-key and customer manage encryption.

Connectivty Security

All communication over internet is via HTTPS.

All communication is secure and encrypted via TLS 1.2 or higher

Compliance Security

Third Party

HIPPA

PCI (Payment Card for Industry for data security)

NIST 800-53

SOC

SOC-2 type II

SIG Lite (SIG Assessment) (Standardized Information Gathering - SIG Questionnaire Tools allow organizations to build, customize, analyze and store vendor questionnaires)

Application Activity Log

History tab provide all historical commands

All details including session id etc can be viewed.

Each query has query id which helps for trouble shooting (no access to data to even SF users)

User Access Audit Log

Login_History family of table function can be use to query login attempts

Query History

Query History is available for 7 days, it can be stored in SF table or external system.

Infrastructure Monitoring

SF users Threat Stack & Sumo Logic to monitor production Infrastructure

Lacework for behavioural Monitoring (use activity/network traffic/binaries)

All alerts are viewed by SF security team.

Penetration Testing

SF performs 7-10 penetration testing per year

Application Penetration Test

Network Penetration Test

Funtional Penetration Test

All logs and findings are tracked to closure

Test results are available with customer under NDA

Data Cloning

-- Cloning a database

```
CREATE DATABASE TIPS_DATABASE_QA
CLONE "TIPS_DATABASE"
COMMENT = 'This is cloned database from Tips Database';
```

-- Create like feature to copy table, but data is not copied

```
CREATE TABLE "TIPS_DATABASE_QA"."LAB".DEVICE_DATA_CREATE_LIKE
LIKE "TIPS_DATABASE_QA"."LAB"."DEVICE_DATA"
COMMENT = 'Create like feature is used which is similar to clone';
```

-- Cloning a file format

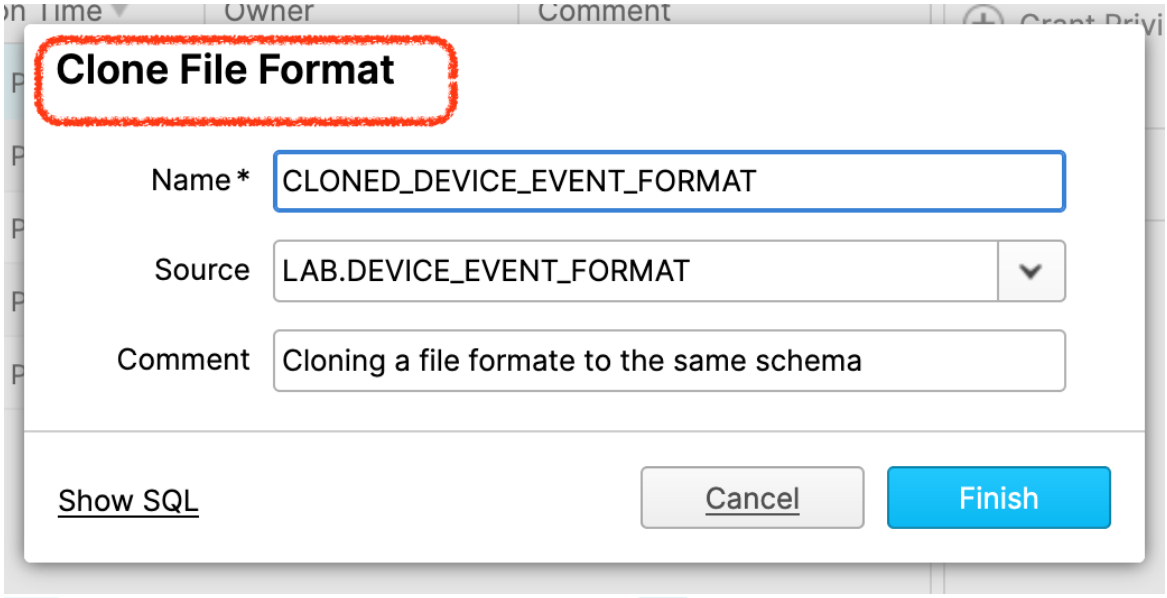
-- img exist

```
CREATE FILE FORMAT "TIPS_DATABASE_QA"."LAB".CLONED_DEVICE_EVENT_FORMAT
CLONE "TIPS_DATABASE_QA"."LAB"."DEVICE_EVENT_FORMAT"
COMMENT = 'Cloning a file formate to the same schema';
```

-- Create a stage via clone.

-- img exist

```
CREATE STAGE "TIPS_DATABASE_QA"."LAB".CLONE_JSON_STAGE
CLONE "TIPS_DATABASE_QA"."LAB"."JSON_STAGE"
COMMENT = 'This is stage cloning for named external stage';
```



Results | Data Preview

Open History

Cloned Database has not yet appered. it will appear when additional storage operation will be performed

View: SNOWFLAKE.ACCOUNT_USAGE.DATABASES | Data | SQL

Filter result...

Columns

Row	DATABASE_ID	DATABASE_NAME	DATABASE_OWN	IS_TRANSIENT	COMMENT	CREATED	LAST_ALTERED	DELETED	RETENTION_TIM
1	6	USDA_NUTRIENT_STDREF	SYSADMIN	NO	USDA Nutrie...	2020-05-02 ...	2020-05-02 ...		1
2	3	DEMO_DB	SYSADMIN	NO	demo databa...	2020-05-02 ...	2020-05-02 ...		1
3	4	SNOWFLAKE_SAMPLE_DATA	ACCOUNTAD...	NO	TPC-H, Open...	2020-05-02 ...	2020-05-02 ...		1
4	8	DB_FOR_SECURITY	SYSADMIN	NO	This is testing	2020-05-11 2...	2020-05-11 2...		1
5	2	UTIL_DB	SYSADMIN	NO	utility databa...	2020-05-02 ...	2020-05-02 ...		1
6	5	MAY022020DB	SYSADMIN	NO	This is sampl...	2020-05-02 ...	2020-05-02 ...		1
7	1	SNOWFLAKE		NO		2020-05-02 ...	2020-05-02 ...		1
8	7	TIPS_DATABASE	SYSADMIN	NO	This is practi...	2020-05-09 ...	2020-05-09 ...		1

Results | Data Preview

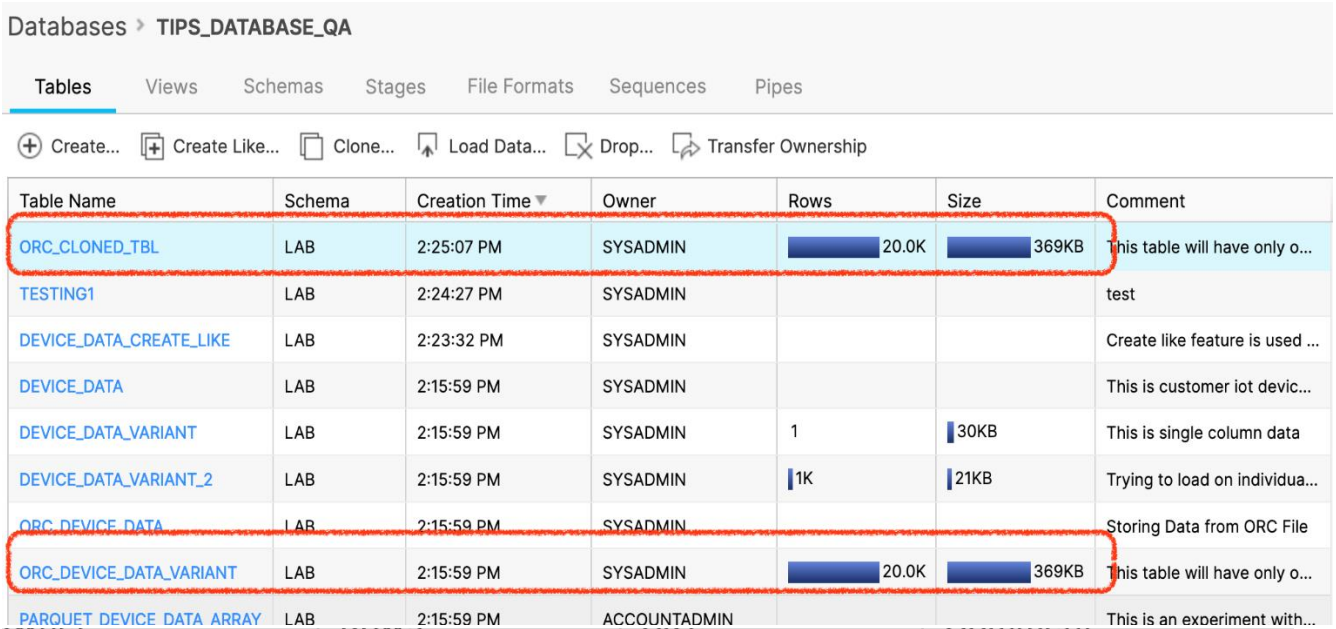
Open History

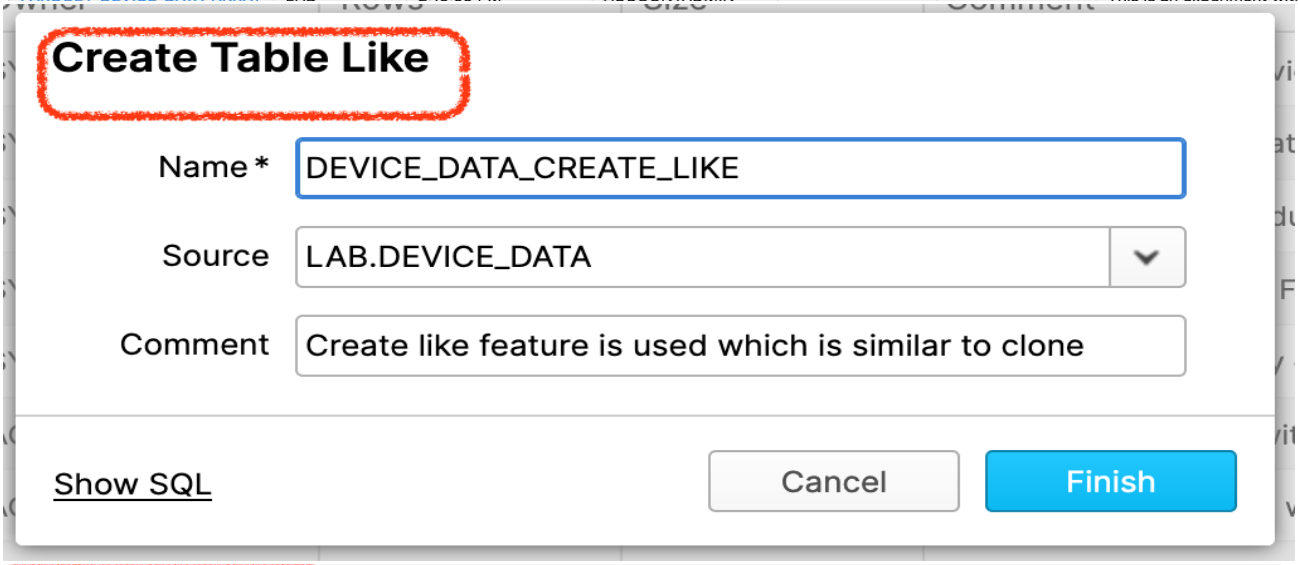
View: TIPS_DATABASE_QA.INFORMATION_SCHEMA.TABLES

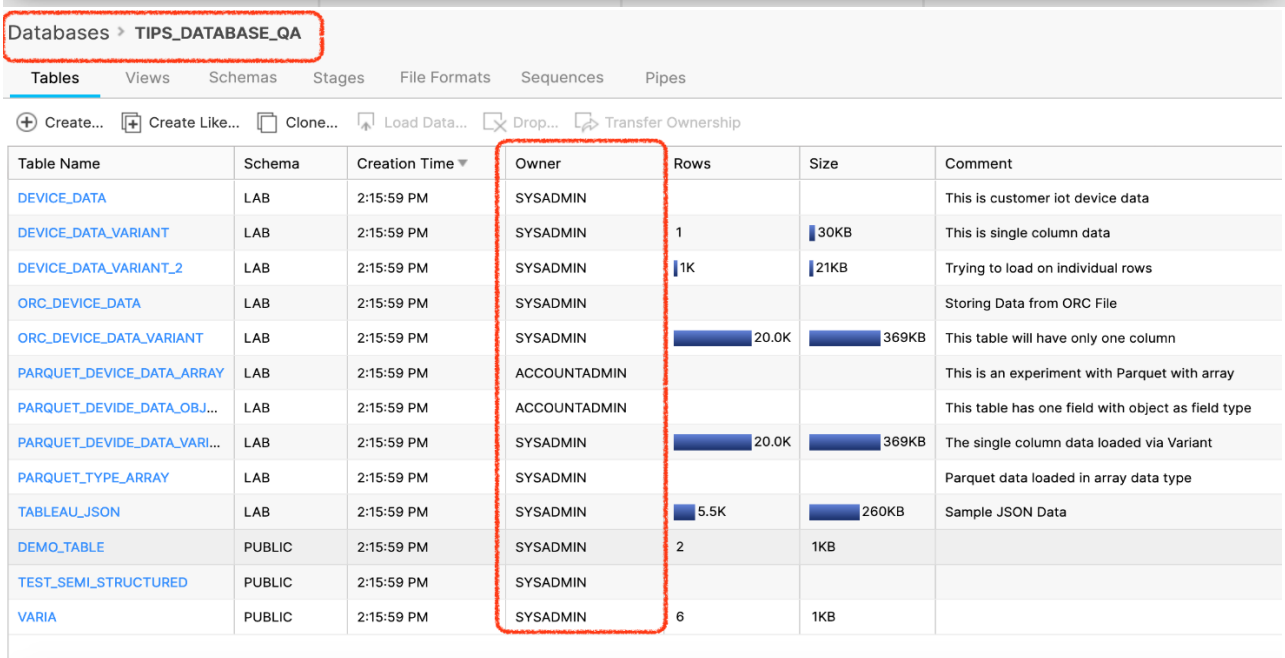
Filter result...

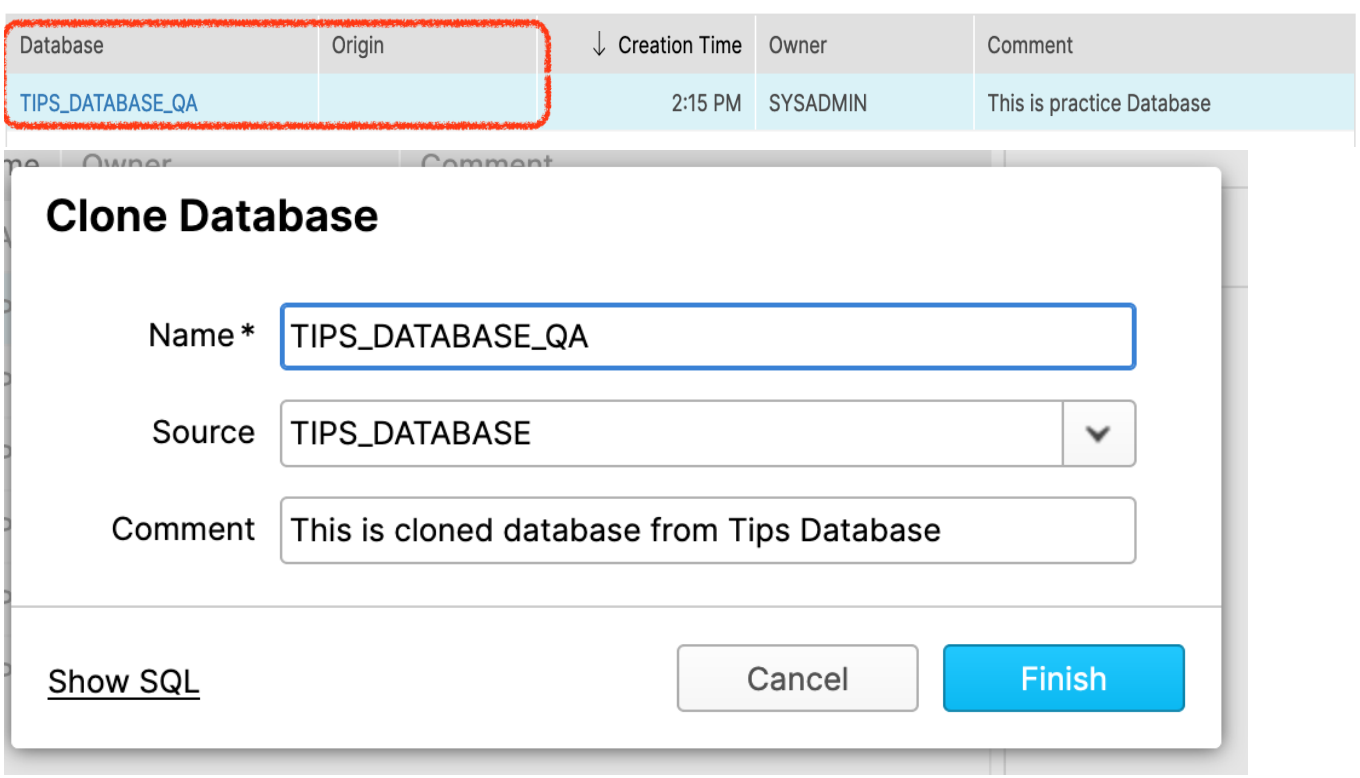
Columns

Row	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_OWNER	TABLE_TYPE	IS_TRANSIENT	CLUSTERING_KI	ROW_COUNT	BYTES	RETENTIO
1	TIPS_DATAB...	LAB	DEVICE_DATA	SYSADMIN	BASE TABLE	NO		0	0	
2	TIPS_DATAB...	LAB	DEVICE_DATA_CREATE_LIKE	SYSADMIN	BASE TABLE	NO		0	0	
3	TIPS_DATAB...	LAB	DEVICE_DATA_VARIANT	SYSADMIN	BASE TABLE	NO		1	30720	
4	TIPS_DATAB...	LAB	DEVICE_DATA_VARIANT_2	SYSADMIN	BASE TABLE	NO		1000	21504	
5	TIPS_DATAB...	LAB	ORC_CLONED_TBL	SYSADMIN	BASE TABLE	NO		19980	377856	
6	TIPS_DATAB...	LAB	ORC_DEVICE_DATA	SYSADMIN	BASE TABLE	NO		0	0	
7	TIPS_DATAB...	LAB	ORC_DEVICE_DATA_VARIANT	SYSADMIN	BASE TABLE	NO		19980	377856	
8	TIPS_DATAB...	LAB	PARQUET_DEVICE_DATA_AR...	ACCOUNTA...	BASE TABLE	NO		0	0	
9	TIPS_DATAB...	LAB	PARQUET_DEVIDE_DATA_OB...	ACCOUNTA...	BASE TABLE	NO		0	0	
10	TIPS_DATAB...	LAB	PARQUET_DEVIDE_DATA_VA...	SYSADMIN	BASE TABLE	NO		19980	377856	
11	TIPS_DATAB...	LAB	PARQUET_TYPE_ARRAY	SYSADMIN	BASE TABLE	NO		0	0	


Databases › TIPS_DATABASE_QA
Tables
Views
Schemas
Stages
File Formats
Sequences
Pipes
Create...
Create Like...
Clone...
Load Data...
Drop...
Transfer Ownership
Table Name | Schema | Creation Time | Owner | Rows | Size | Comment
ORC_CLONED_TBL | LAB | 2:25:07 PM | SYSADMIN | 20.0K | 369KB | This table will have only o...
TESTING1 | LAB | 2:24:27 PM | SYSADMIN | | | test
DEVICE_DATA_CREATE_LIKE | LAB | 2:23:32 PM | SYSADMIN | | | Create like feature is used ...
DEVICE_DATA | LAB | 2:15:59 PM | SYSADMIN | | | This is customer iot devic...
DEVICE_DATA_VARIANT | LAB | 2:15:59 PM | SYSADMIN | 1 | 30KB | This is single column data
DEVICE_DATA_VARIANT_2 | LAB | 2:15:59 PM | SYSADMIN | 1K | 21KB | Trying to load on individua...
ORC_DEVICE_DATA | LAB | 2:15:59 PM | SYSADMIN | | | Storing Data from ORC File
ORC_DEVICE_DATA_VARIANT | LAB | 2:15:59 PM | SYSADMIN | 20.0K | 369KB | This table will have only o...
PARQUET_DEVICE_DATA_ARRAY | LAB | 2:15:59 PM | ACCOUNTADMIN | | | This is an experiment with...


Create Table Like
Name *
DEVICE_DATA_CREATE_LIKE
Source
LAB.DEVICE_DATA
Comment
Create like feature is used which is similar to clone
Show SQL
Cancel
Finish


Databases › TIPS_DATABASE_QA
Tables
Views
Schemas
Stages
File Formats
Sequences
Pipes
Create...
Create Like...
Clone...
Load Data...
Drop...
Transfer Ownership
Table Name | Schema | Creation Time | Owner | Rows | Size | Comment
DEVICE_DATA | LAB | 2:15:59 PM | SYSADMIN | | | This is customer iot device data
DEVICE_DATA_VARIANT | LAB | 2:15:59 PM | SYSADMIN | 1 | 30KB | This is single column data
DEVICE_DATA_VARIANT_2 | LAB | 2:15:59 PM | SYSADMIN | 1K | 21KB | Trying to load on individual rows
ORC_DEVICE_DATA | LAB | 2:15:59 PM | SYSADMIN | | | Storing Data from ORC File
ORC_DEVICE_DATA_VARIANT | LAB | 2:15:59 PM | SYSADMIN | 20.0K | 369KB | This table will have only one column
PARQUET_DEVICE_DATA_ARRAY | LAB | 2:15:59 PM | ACCOUNTADMIN | | | This is an experiment with Parquet with array
PARQUET_DEVIDE_DATA_OBJ... | LAB | 2:15:59 PM | ACCOUNTADMIN | | | This table has one field with object as field type
PARQUET_DEVIDE_DATA_VARI... | LAB | 2:15:59 PM | SYSADMIN | 20.0K | 369KB | The single column data loaded via Variant
PARQUET_TYPE_ARRAY | LAB | 2:15:59 PM | SYSADMIN | | | Parquet data loaded in array data type
TABLEAU_JSON | LAB | 2:15:59 PM | SYSADMIN | 5.5K | 260KB | Sample JSON Data
DEMO_TABLE | PUBLIC | 2:15:59 PM | SYSADMIN | 2 | 1KB |
TEST_SEMI_STRUCTURED | PUBLIC | 2:15:59 PM | SYSADMIN | | |
VARIA | PUBLIC | 2:15:59 PM | SYSADMIN | 6 | 1KB |


Database
Origin
Creation Time
Owner
Comment
TIPS_DATABASE_QA
2:15 PM
SYSADMIN
This is practice Database
Clone Database
Name *
TIPS_DATABASE_QA
Source
TIPS_DATABASE
Comment
This is cloned database from Tips Database
Show SQL
Cancel
Finish

Account Usage

- SNOWFLAKE is a system-defined, read-only shared database, provided by Snowflake

- The database is automatically imported into each account from a share named ACCOUNT_USAGE

- The SNOWFLAKE database is an example of Snowflake utilizing Secure Data Sharing to provide object metadata and other usage metrics for your account.

- The SNOWFLAKE database contains three schemas (also read-only). Each schema contains a set of views (refer images)

- ACCOUNT_USAGE: Views that display object metadata and usage metrics for your account.

- Diff in ACCOUNT_USAGE vs INFORMATION_SCHEMA

- Records for dropped objects included in each view.

- Longer retention time for historical usage data.

- Data latency.

- By default, only account administrators (users with the ACCOUNTADMIN role) can access the SNOWFLAKE database and schemas within the database, or perform queries on the views

-- Enabling Account Usage for Other Roles

```
USE ROLE ACCOUNTADMIN;

GRANT IMPORTED PRIVILEGES ON DATABASE snowflake TO ROLE SYSADMIN;

GRANT IMPORTED PRIVILEGES ON DATABASE snowflake TO ROLE customrole1;
```

USE ROLE customrole1;

```
SELECT * FROM snowflake.account_usage.databases;
```

-- Sample Queries

-- Average number of seconds between failed login attempts by user (month-to-date):

-- refer image for result and cost of the query

```
USE ROLE ACCOUNTADMIN;
select user_name,
      count(*) as failed_logins,
      avg(seconds_between_login_attempts) as average_seconds_between_login_attempts
from (
     select user_name,
           timediff(seconds, event_timestamp, lead(event_timestamp)
              over(partition by user_name order by event_timestamp)) as
seconds_between_login_attempts
     from "SNOWFLAKE"."ACCOUNT_USAGE"."LOGIN_HISTORY"
     where event_timestamp > date_trunc(month, current_date)
     and is_success = 'NO'
    )
group by 1
order by 3;
```

--Failed logins by user (month-to-date):

-- refer image for result

```
select user_name,
      sum(iff(is_success = 'NO', 1, 0)) as failed_logins,
      count(*) as logins,
      sum(iff(is_success = 'NO', 1, 0)) / nullif(count(*), 0) as login_failure_rate
from "SNOWFLAKE"."ACCOUNT_USAGE"."LOGIN_HISTORY"
where event_timestamp > date_trunc(month, current_date)
group by 1
order by 4 desc;
```

-- Failed logins by user and connecting client (month-to-date):

```
select reported_client_type,
    user_name,
    sum(iff(is_success = 'NO', 1, 0)) as failed_logins,
    count(*) as logins,
    sum(iff(is_success = 'NO', 1, 0)) / nullif(count(*), 0) as login_failure_rate
from "SNOWFLAKE"."ACCOUNT_USAGE"."LOGIN_HISTORY"
where event_timestamp > date_trunc(month, current_date)
group by 1,2
order by 5 desc;
```

-- Examples: Warehouse Credit Usage

-- Credits used by each warehouse in your account (month-to-date):

-- refer image for result

```
select warehouse_name,
    sum(credits_used) as total_credits_used
from "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY"
where start_time >= date_trunc(month, current_date)
group by 1
order by 2 desc;
```

-- Credits used over time by each warehouse in your account (month-to-date):

-- refer image for result

```
select start_time::date as usage_date,
    warehouse_name,
    sum(credits_used) as total_credits_used
from "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY"
where start_time >= date_trunc(month, current_date)
group by 1,2
```

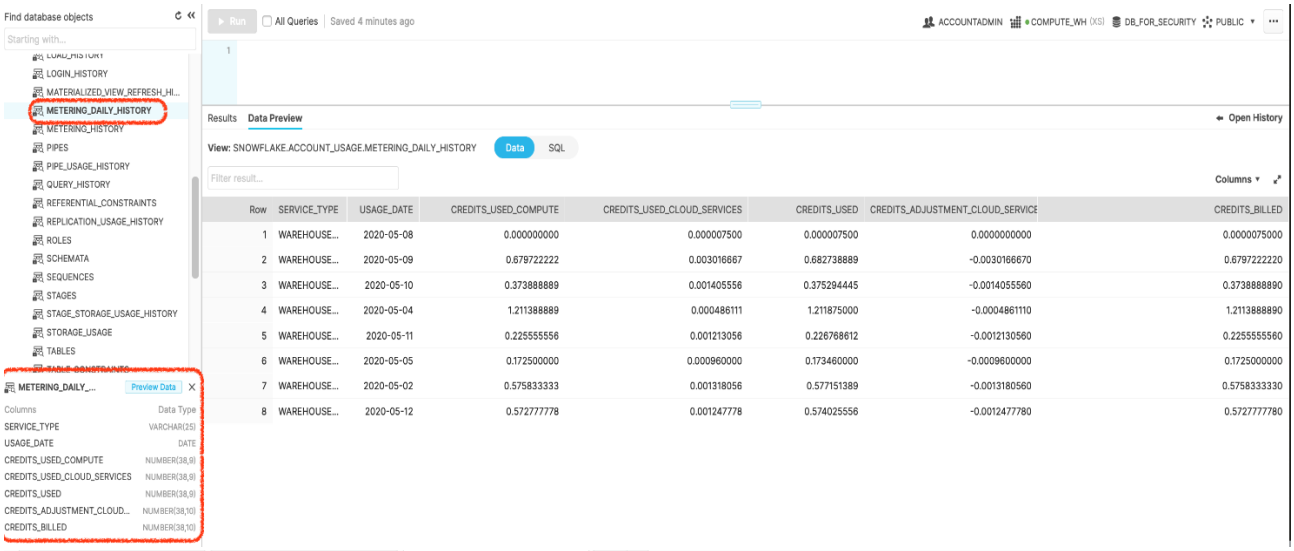
```
order by 2,1;


-- ----------------------------
-- Examples: Data Storage Usage
-------------------------------

-- Billable terabytes stored in your account over time:
select date_trunc(month, usage_date) as usage_month
  , avg(storage_bytes + stage_bytes + failsafe_bytes) / power(1024, 4) as billable_tb
from "SNOWFLAKE"."ACCOUNT_USAGE"."STORAGE_USAGE"
group by 1
order by 1;


-------------------------------
-- Examples: User Query Totals and Execution Times
-------------------------------
-- Total jobs executed in your account (month-to-date):

select count(*) as number_of_jobs
from query_history
where start_time >= date_trunc(month, current_date);

-- Total jobs executed by each warehouse in your account (month-to-date):

select warehouse_name,
count(*) as number_of_jobs
from query_history
where start_time >= date_trunc(month, current_date)
group by 1
order by 2 desc;
```

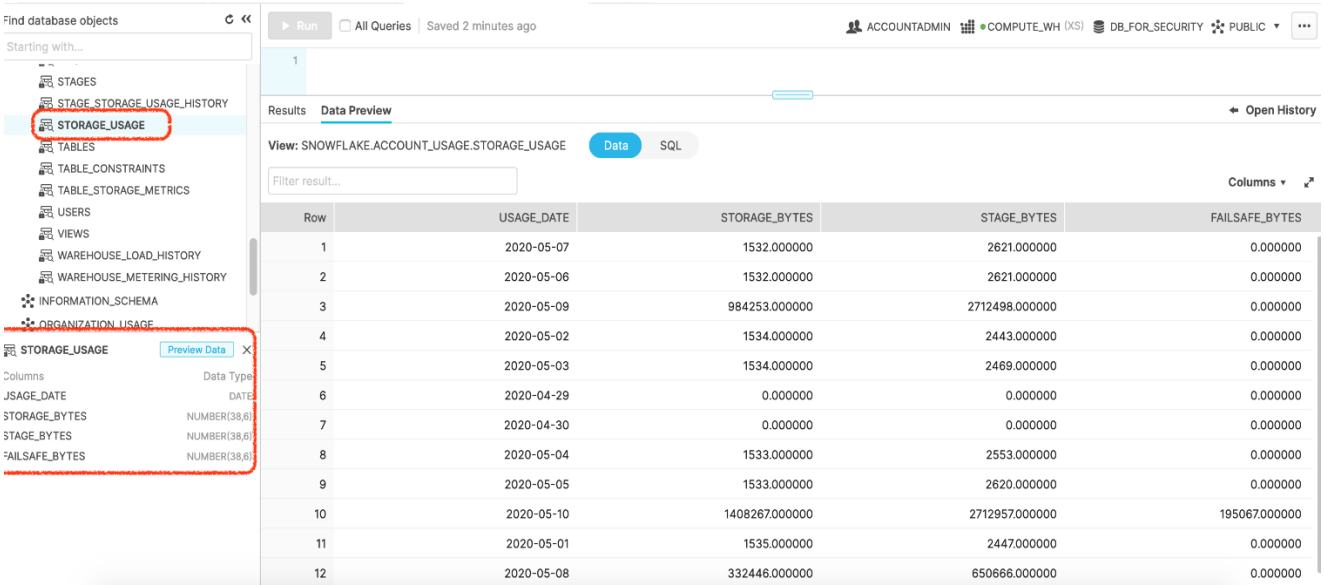
--Average query execution time by user (month-to-date):

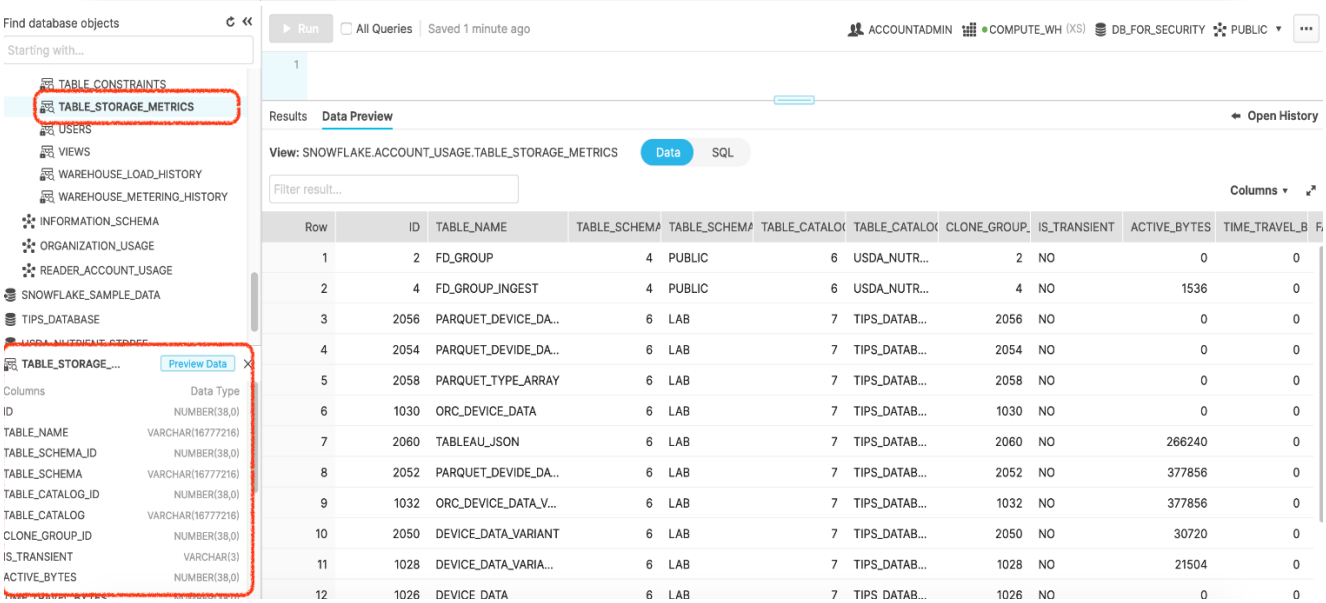
```
select user_name,
avg(execution_time) as average_execution_time
from query_history
where start_time >= date_trunc(month, current_date)
group by 1
order by 2 desc;
```

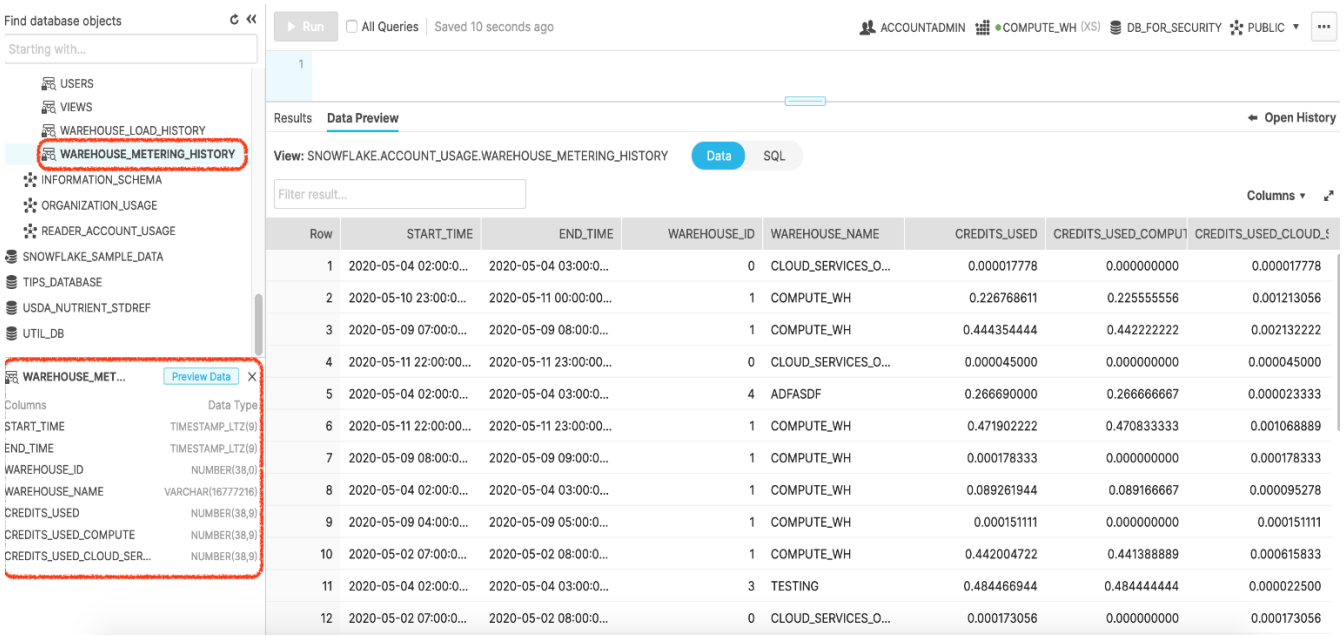
--Average query execution time by query type and warehouse size (month-to-date):

```
select query_type,
warehouse_size,
avg(execution_time) as average_execution_time
from query_history
where start_time >= date_trunc(month, current_date)
group by 1,2
order by 3 desc;
```


Find database objects
METERING_DAILY_HISTORY
Run
All Queries
Saved 4 minutes ago
ACCOUNTADMIN
COMPUTE_WH (XS)
DB_FOR_SECURITY
PUBLIC
Results
Data Preview
Open History
View: SNOWFLAKE.ACCOUNT_USAGE.METERING_DAILY_HISTORY
Data
SQL
Filter result...
Columns
Row SERVICE_TYPE USAGE_DATE CREDITS_USED_COMPUTE CREDITS_USED_CLOUD_SERVICES CREDITS_USED CREDITS_ADJUSTMENT_CLOUD_SERVICE CREDITS_BILLED
1 WAREHOUSE... 2020-05-08 0.000000000 0.000007500 0.000007500 0.0000000000 0.0000075000
2 WAREHOUSE... 2020-05-09 0.679722222 0.003016667 0.682738889 -0.0030166670 0.6797222220
3 WAREHOUSE... 2020-05-10 0.373888889 0.001405556 0.375294445 -0.0014055560 0.3738888890
4 WAREHOUSE... 2020-05-04 1.211388889 0.000486111 1.211875000 -0.0004861110 1.2113888890
5 WAREHOUSE... 2020-05-11 0.225555556 0.001213056 0.226768612 -0.0012130560 0.2255555560
6 WAREHOUSE... 2020-05-05 0.172500000 0.000960000 0.173460000 -0.0009600000 0.1725000000
7 WAREHOUSE... 2020-05-02 0.575833333 0.001318056 0.577151389 -0.0013180560 0.5758333330
8 WAREHOUSE... 2020-05-12 0.572777778 0.001247778 0.574025556 -0.0012477780 0.5727777780
METERING_DAILY_...
Preview Data
Columns
Data Type
SERVICE_TYPE VARCHAR(25)
USAGE_DATE DATE
CREDITS_USED_COMPUTE NUMBER(38,9)
CREDITS_USED_CLOUD_SERVICES NUMBER(38,9)
CREDITS_USED NUMBER(38,9)
CREDITS_ADJUSTMENT_CLOUD... NUMBER(38,10)
CREDITS_BILLED NUMBER(38,10)


Find database objects
STORAGE_USAGE
Run
All Queries
Saved 2 minutes ago
ACCOUNTADMIN
COMPUTE_WH (XS)
DB_FOR_SECURITY
PUBLIC
Results
Data Preview
Open History
View: SNOWFLAKE.ACCOUNT_USAGE.STORAGE_USAGE
Data
SQL
Filter result...
Columns
Row USAGE_DATE STORAGE_BYTES STAGE_BYTES FAILSAFE_BYTES
1 2020-05-07 1532.000000 2621.000000 0.000000
2 2020-05-06 1532.000000 2621.000000 0.000000
3 2020-05-09 984253.000000 2712498.000000 0.000000
4 2020-05-02 1534.000000 2443.000000 0.000000
5 2020-05-03 1534.000000 2469.000000 0.000000
6 2020-04-29 0.000000 0.000000 0.000000
7 2020-04-30 0.000000 0.000000 0.000000
8 2020-05-04 1533.000000 2553.000000 0.000000
9 2020-05-05 1533.000000 2620.000000 0.000000
10 2020-05-10 1408267.000000 2712957.000000 195067.000000
11 2020-05-01 1535.000000 2447.000000 0.000000
12 2020-05-08 332446.000000 650666.000000 0.000000
STORAGE_USAGE
Preview Data
Columns
Data Type
USAGE_DATE DATE
STORAGE_BYTES NUMBER(38,6)
STAGE_BYTES NUMBER(38,6)
FAILSAFE_BYTES NUMBER(38,6)


Find database objects
TABLE_STORAGE_METRICS
Run
All Queries
Saved 1 minute ago
ACCOUNTADMIN
COMPUTE_WH (XS)
DB_FOR_SECURITY
PUBLIC
Results
Data Preview
Open History
View: SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS
Data
SQL
Filter result...
Columns
Row ID TABLE_NAME TABLE_SCHEMA TABLE_SCHEMA TABLE_CATALO TABLE_CATALO CLONE_GROUP_ IS_TRANSIENT ACTIVE_BYTES TIME_TRAVEL_B
1 2 FD_GROUP 4 PUBLIC 6 USDA_NUTR... 2 NO 0 0
2 4 FD_GROUP_INGEST 4 PUBLIC 6 USDA_NUTR... 4 NO 1536 0
3 2056 PARQUET_DEVICE_DA... 6 LAB 7 TIPS_DATAB... 2056 NO 0 0
4 2054 PARQUET_DEVIDE_DA... 6 LAB 7 TIPS_DATAB... 2054 NO 0 0
5 2058 PARQUET_TYPE_ARRAY 6 LAB 7 TIPS_DATAB... 2058 NO 0 0
6 1030 ORC_DEVICE_DATA 6 LAB 7 TIPS_DATAB... 1030 NO 0 0
7 2060 TABLEAU_JSON 6 LAB 7 TIPS_DATAB... 2060 NO 266240 0
8 2052 PARQUET_DEVIDE_DA... 6 LAB 7 TIPS_DATAB... 2052 NO 377856 0
9 1032 ORC_DEVICE_DATA_V... 6 LAB 7 TIPS_DATAB... 1032 NO 377856 0
10 2050 DEVICE_DATA_VARIANT 6 LAB 7 TIPS_DATAB... 2050 NO 30720 0
11 1028 DEVICE_DATA_VARIA... 6 LAB 7 TIPS_DATAB... 1028 NO 21504 0
12 1026 DEVICE_DATA 6 LAB 7 TIPS_DATAB... 1026 NO 0 0
TABLE_STORAGE_...
Preview Data
Columns
Data Type
ID NUMBER(38,0)
TABLE_NAME VARCHAR(16777216)
TABLE_SCHEMA_ID NUMBER(38,0)
TABLE_SCHEMA VARCHAR(16777216)
TABLE_CATALOG_ID NUMBER(38,0)
TABLE_CATALOG VARCHAR(16777216)
CLONE_GROUP_ID NUMBER(38,0)
IS_TRANSIENT VARCHAR(3)
ACTIVE_BYTES NUMBER(38,0)




```
Run | All Queries | Saved 13 seconds ago | ACCOUNTADMIN | COMPUTE_WH (XS) | DB_FOR_SECURITY | PUBLIC
1 select date_trunc(month, usage_date) as usage_month
2   , avg(storage_bytes + stage_bytes + failsafe_bytes) / power(1024, 4) as billable_tb
3 from "SNOWFLAKE"."ACCOUNT_USAGE"."STORAGE_USAGE"
4 group by 1
5 order by 1;
```

Results | Data Preview | Open History

✔ Query ID | SQL | 895ms | 2 rows

Filter result... | Copy | Columns

Row	USAGE_MONTH	BILLABLE_TB
1	2020-04-01	0
2	2020-05-01	1.10333291e-06

```
Run | All Queries | Saved 25 seconds ago | ACCOUNTADMIN | COMPUTE_WH (XS) | DB_FOR_SECURITY | PUBLIC
1 select start_time::date as usage_date,
2        warehouse_name,
3        sum(credits_used) as total_credits_used
4 from "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY"
5 where start_time >= date_trunc(month, current_date)
6 group by 1,2
7 order by 2,1;
```

Results | Data Preview | Open History

✔ Query ID | SQL | 1.11s | 15 rows

Filter result... | Copy | Columns

Row	USAGE_DATE	WAREHOUSE_NAME	TOTAL_CREDITS_USED
1	2020-05-04	ADFASDF	0.266690000
2	2020-05-02	CLOUD_SERVICES_ONLY	0.000702223
3	2020-05-04	CLOUD_SERVICES_ONLY	0.000017778
4	2020-05-05	CLOUD_SERVICES_ONLY	0.000006111
5	2020-05-11	CLOUD_SERVICES_ONLY	0.000045000
6	2020-05-04	COMPUTE_2CREDIT	0.371451111
7	2020-05-05	COMPUTE_2CREDIT	0.000005556
8	2020-05-02	COMPUTE_WH	0.576449166
9	2020-05-04	COMPUTE_WH	0.089261944

```sql
select warehouse_name,
       sum(credits_used) as total_credits_used
from "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY"
where start_time >= date_trunc(month, current_date)
group by 1
order by 2 desc;
```

Results | Data Preview — Open History

Query ID | SQL — 1.36s — 5 rows

Filter result... — Copy — Columns

Row	WAREHOUSE_NAME	TOTAL_CREDITS_USED
1	COMPUTE_WH	2.595891664
2	TESTING	0.484466944
3	COMPUTE_2CREDIT	0.371456667
4	ADFASDF	0.266690000
5	CLOUD_SERVICES_ONLY	0.000771112

History — Las

Hide Filters | View SQL | Abort...

Add a filter to customize your results.

☐ Include client-generated statements

☐ Include queries executed by user tasks

Status	Incident	Query ID	SQL Text	User	Warehouse	Size
✓		01942c60-...	select user_name, count(*) as failed_logins, avg(seconds_between_login_attempts) as average...	USER01	COMPUTE_...	X-Small
✓		01942c43-...	select warehouse_name, sum(credits_used) as total_credits_us...			
✓		01942c43-...	use role accountadmin;			
✗		01942c42-...	select warehouse_name, sum(credits_used) as total_credits_us...			

```sql
select user_name, count(*) as failed_logins, avg(seconds_between_login_attempts) as average_seconds_between_login_attempts from ( select user_name, timediff(seconds, event_timestamp, lead(event_timestamp) over(partition by user_name order by event_timestamp)) as seconds_between_login_attempts from "SNOWFLAKE"."ACCOUNT_USAGE"."LOGIN_HISTORY" where event_timestamp > date_trunc(month, current_date) and is_success = 'NO' ) group by 1 order by 3;
```

Find database objects

Starting with...

- DEMO_DB
- MAY022020DB
- SNOWFLAKE
  - ACCOUNT_USAGE
    - No Tables in this Schema
    - Views
      - AUTOMATIC_CLUSTERING_HISTORY
      - COLUMNS
      - COPY_HISTORY
      - DATABASES
      - DATABASE_STORAGE_USAGE_HIST...
      - DATA_TRANSFER_HISTORY
      - FILE_FORMATS
      - FUNCTIONS
      - GRANTS_TO_ROLES
      - GRANTS_TO_USERS
      - LOAD_HISTORY
      - LOGIN_HISTORY
      - MATERIALIZED_VIEW_REFRESH_HIS...
      - METERING_DAILY_HISTORY
      - METERING_HISTORY

Run | ☐ All Queries | Saved 1 minute ago — ACCOUNTADMIN — COMPUTE_WH (XS)

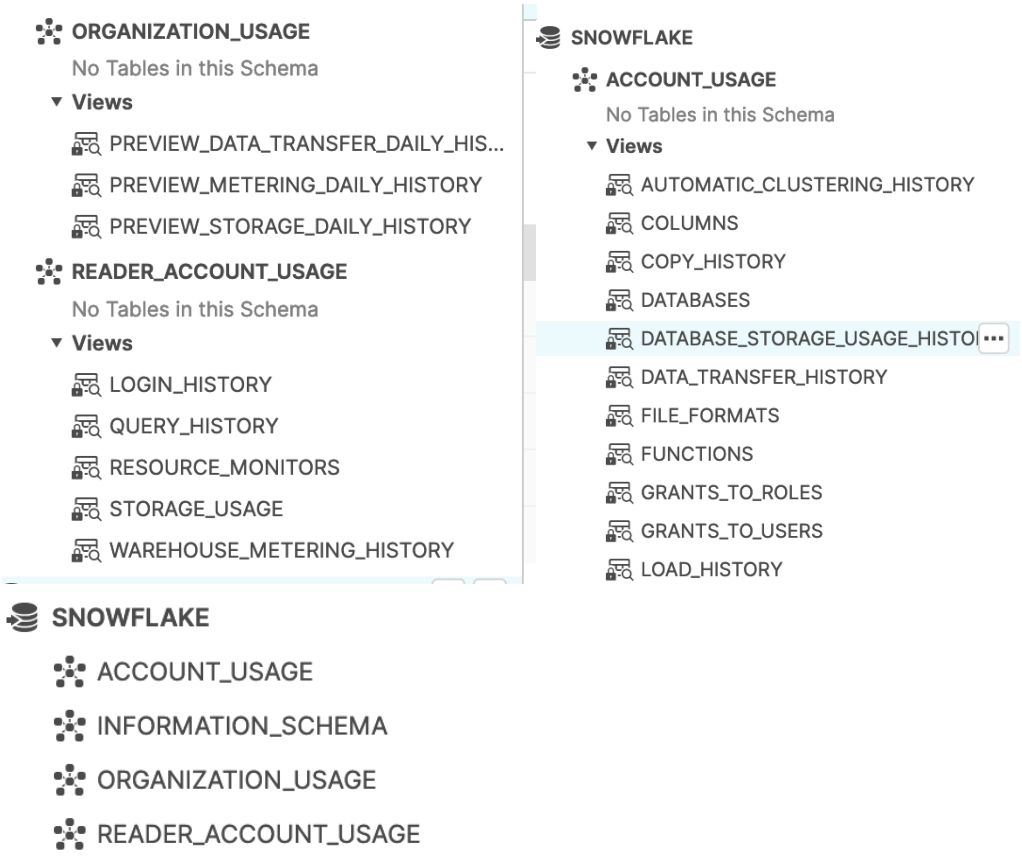
```sql
select user_name,
       count(*) as failed_logins,
       avg(seconds_between_login_attempts) as average_seconds_between_login_attempts
from (
     select user_name,
            timediff(seconds, event_timestamp, lead(event_timestamp)
                over(partition by user_name order by event_timestamp)) as seconds_between_login_attempts
     from "SNOWFLAKE"."ACCOUNT_USAGE"."LOGIN_HISTORY"
     where event_timestamp > date_trunc(month, current_date)
     and is_success = 'NO'
    )
group by 1
order by 3;
```

Total Duration	1.43s
Queuing	100ms
Compilation	521ms
Execution	805ms
Compiling SQL	521ms
Executing (serial)	2ms
Queued (resuming warehouse partition)	100ms
Wait process group	203ms
Executing (warehouse)	534ms
Receiving request from client	24ms

Results

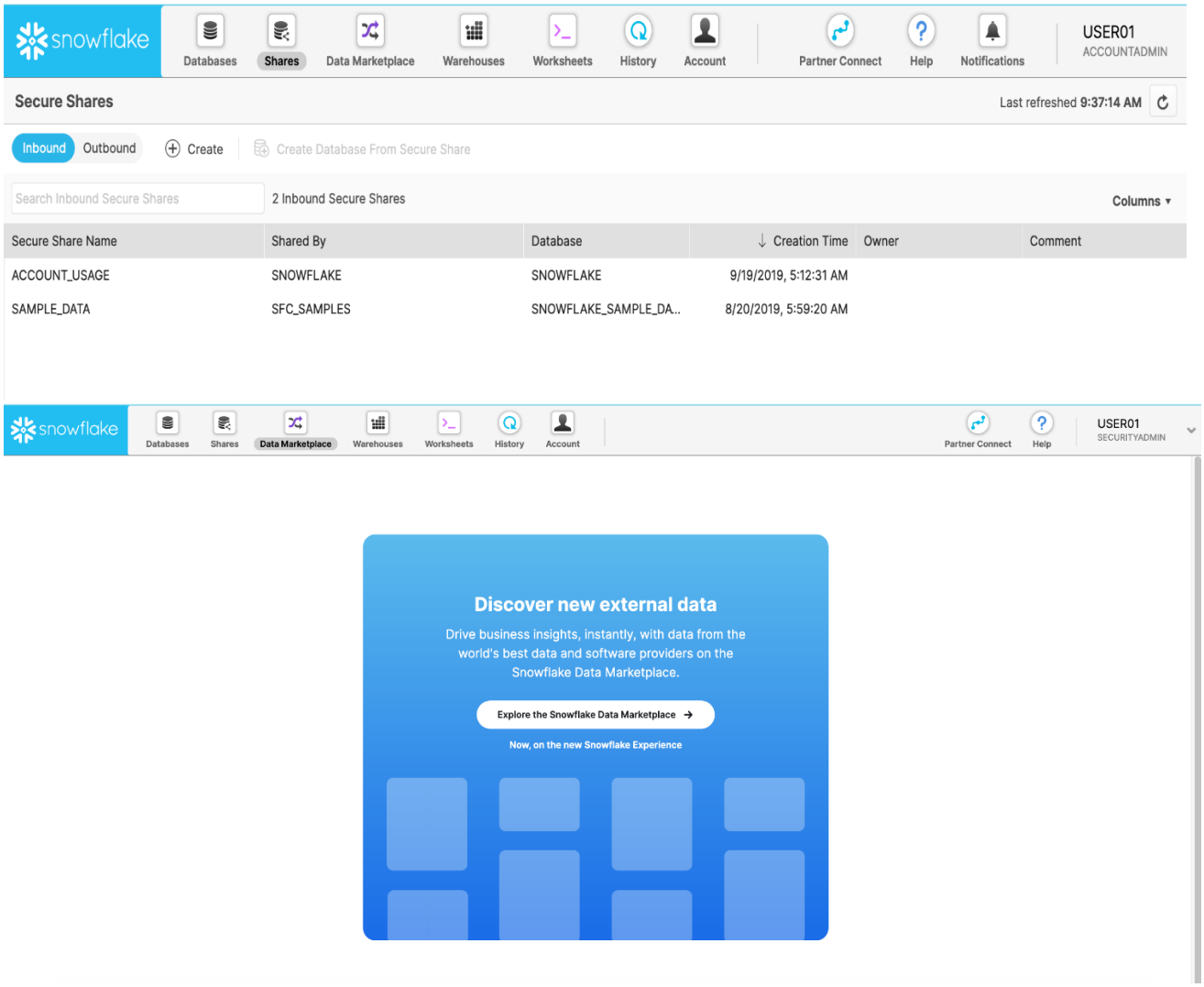
Query ID | SQL — 1.43s — 0 rows

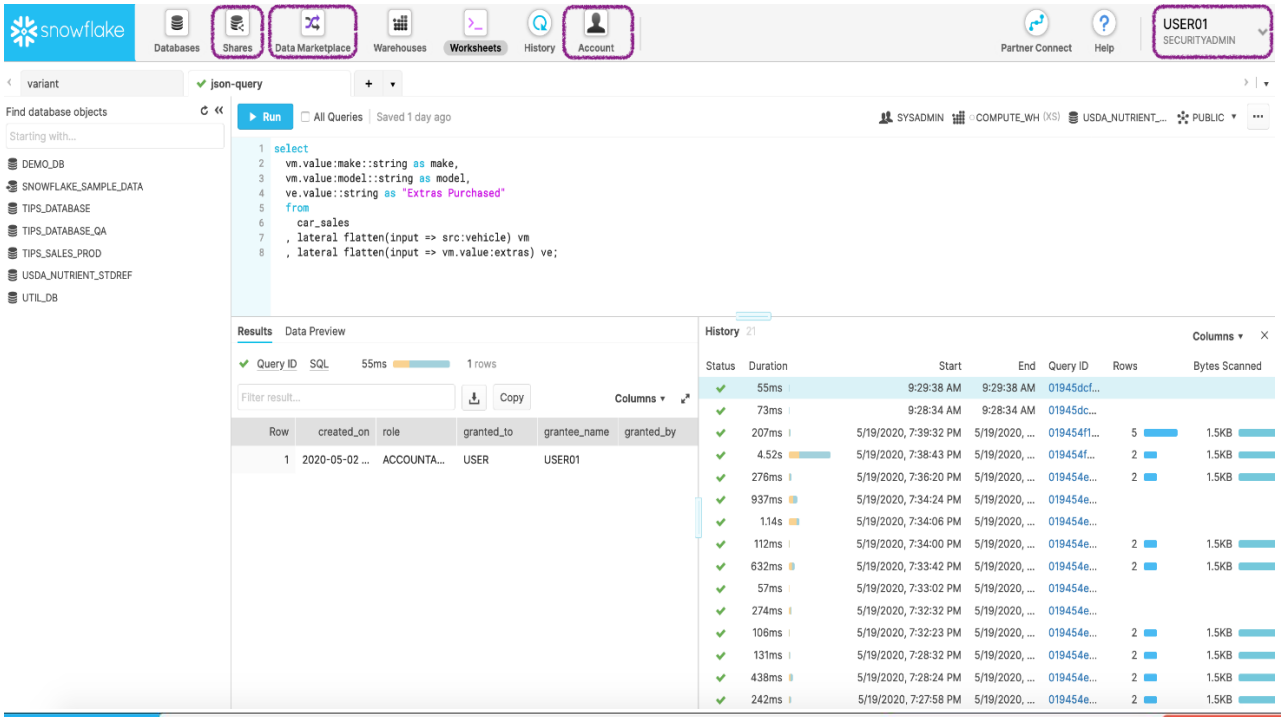
Filter result — Copy

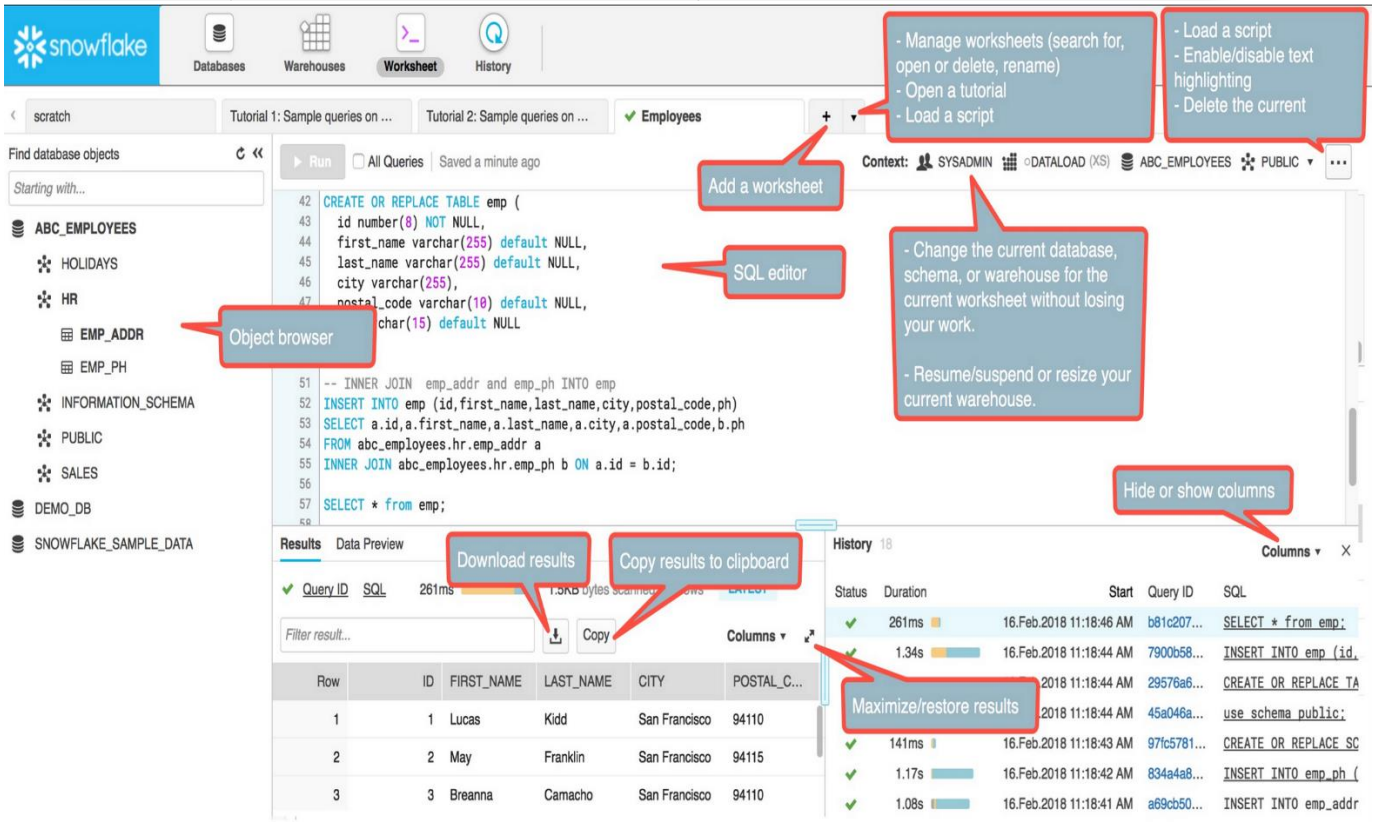

ORGANIZATION_USAGE
No Tables in this Schema
Views
PREVIEW_DATA_TRANSFER_DAILY_HIS...
PREVIEW_METERING_DAILY_HISTORY
PREVIEW_STORAGE_DAILY_HISTORY
READER_ACCOUNT_USAGE
No Tables in this Schema
Views
LOGIN_HISTORY
QUERY_HISTORY
RESOURCE_MONITORS
STORAGE_USAGE
WAREHOUSE_METERING_HISTORY
SNOWFLAKE
ACCOUNT_USAGE
No Tables in this Schema
Views
AUTOMATIC_CLUSTERING_HISTORY
COLUMNS
COPY_HISTORY
DATABASES
DATABASE_STORAGE_USAGE_HISTO
DATA_TRANSFER_HISTORY
FILE_FORMATS
FUNCTIONS
GRANTS_TO_ROLES
GRANTS_TO_USERS
LOAD_HISTORY

SNOWFLAKE
ACCOUNT_USAGE
INFORMATION_SCHEMA
ORGANIZATION_USAGE
READER_ACCOUNT_USAGE

WebUI


snowflake
Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Account
Partner Connect
Help
Notifications
USER01
ACCOUNTADMIN
Secure Shares
Last refreshed 9:37:14 AM
Inbound
Outbound
Create
Create Database From Secure Share
Search Inbound Secure Shares
2 Inbound Secure Shares
Columns
Secure Share Name
Shared By
Database
Creation Time
Owner
Comment
ACCOUNT_USAGE
SNOWFLAKE
SNOWFLAKE
9/19/2019, 5:12:31 AM
SAMPLE_DATA
SFC_SAMPLES
SNOWFLAKE_SAMPLE_DA...
8/20/2019, 5:59:20 AM
snowflake
Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Account
Partner Connect
Help
USER01
SECURITYADMIN
Discover new external data
Drive business insights, instantly, with data from the world's best data and software providers on the Snowflake Data Marketplace.
Explore the Snowflake Data Marketplace
Now, on the new Snowflake Experience


snowflake
Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Account
Partner Connect
Help
USER01
SECURITYADMIN
variant
json-query
Find database objects
Starting with...
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
TIPS_DATABASE
TIPS_DATABASE_QA
TIPS_SALES_PROD
USDA_NUTRIENT_STDREF
UTIL_DB
Run
All Queries
Saved 1 day ago
SYSADMIN
COMPUTE_WH (XS)
USDA_NUTRIENT_...
PUBLIC
select
vm.value:make::string as make,
vm.value:model::string as model,
ve.value::string as "Extras Purchased"
from
car_sales
, lateral flatten(input => src:vehicle) vm
, lateral flatten(input => vm.value:extras) ve;
Results
Data Preview
Query ID
SQL
55ms
1 rows
Filter result...
Copy
Columns
Row
created_on
role
granted_to
grantee_name
granted_by
1
2020-05-02 ...
ACCOUNTA...
USER
USER01
History
Status
Duration
Start
End
Query ID
Rows
Bytes Scanned


snowflake
Databases
Warehouses
Worksheet
History
- Manage worksheets (search for, open or delete, rename)
- Open a tutorial
- Load a script
- Load a script
- Enable/disable text highlighting
- Delete the current
scratch
Tutorial 1: Sample queries on ...
Tutorial 2: Sample queries on ...
Employees
Add a worksheet
Find database objects
Starting with...
ABC_EMPLOYEES
HOLIDAYS
HR
EMP_ADDR
EMP_PH
INFORMATION_SCHEMA
PUBLIC
SALES
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
Object browser
Run
All Queries
Saved a minute ago
Context: SYSADMIN DATALOAD (XS) ABC_EMPLOYEES PUBLIC
SQL editor
- Change the current database, schema, or warehouse for the current worksheet without losing your work.
- Resume/suspend or resize your current warehouse.
CREATE OR REPLACE TABLE emp (
id number(8) NOT NULL,
first_name varchar(255) default NULL,
last_name varchar(255) default NULL,
city varchar(255),
postal_code varchar(10) default NULL,
char(15) default NULL
-- INNER JOIN emp_addr and emp_ph INTO emp
INSERT INTO emp (id,first_name,last_name,city,postal_code,ph)
SELECT a.id,a.first_name,a.last_name,a.city,a.postal_code,b.ph
FROM abc_employees.hr.emp_addr a
INNER JOIN abc_employees.hr.emp_ph b ON a.id = b.id;
SELECT * from emp;
Hide or show columns
Results
Data Preview
Download results
Copy results to clipboard
Query ID
SQL
261ms
Filter result...
Copy
Columns
Maximize/restore results
Row
ID
FIRST_NAME
LAST_NAME
CITY
POSTAL_C...
1
1
Lucas
Kidd
San Francisco
94110
2
2
May
Franklin
San Francisco
94115
3
3
Breanna
Camacho
San Francisco
94110
History
Status
Duration
Start
Query ID
SQL
261ms
16.Feb.2018 11:18:46 AM
b81c207...
SELECT * from emp;
1.34s
16.Feb.2018 11:18:44 AM
7900b58...
INSERT INTO emp (id,
16.Feb.2018 11:18:44 AM
29576a6...
CREATE OR REPLACE TA
16.Feb.2018 11:18:44 AM
45a046a...
use schema public;
141ms
16.Feb.2018 11:18:43 AM
97fc5781...
CREATE OR REPLACE SC
1.17s
16.Feb.2018 11:18:42 AM
834a4a8...
INSERT INTO emp_ph (
1.08s
16.Feb.2018 11:18:41 AM
a69cb50...
INSERT INTO emp_addr

User Role Creation

--create a user (Blr@4950$)

```
CREATE USER tips_jr_analyst
PASSWORD = '**********'
COMMENT = 'Tips Junior Analyst to run queries'
MUST_CHANGE_PASSWORD = TRUE;
```

-- with advance parameters

```
CREATE USER tips_jr_analyst
PASSWORD = '**********'
COMMENT = 'Tips Junior Analyst to run queries'
LOGIN_NAME = 'TipsJrAnalyst'
DISPLAY_NAME = 'Junior Analyst'
FIRST_NAME = 'Junior'
LAST_NAME = 'Analyst'
EMAIL = 'tips.analyst@toppertips.com'
MUST_CHANGE_PASSWORD = TRUE;
```

-- With advance params & default warehouse details

-- also granting role

```
CREATE USER tips_jr_analyst
PASSWORD = '**********'
COMMENT = 'Tips Junior Analyst to run queries'
LOGIN_NAME = 'TipsJrAnalyst'
DISPLAY_NAME = 'Junior Analyst'
FIRST_NAME = 'Junior'
LAST_NAME = 'Analyst'
EMAIL = 'tips.analyst@toppertips.com'
```

DEFAULT_ROLE = "USERADMIN"

DEFAULT_WAREHOUSE = 'COMPUTE_WH'

DEFAULT_NAMESPACE = 'TIPS_DB'

MUST_CHANGE_PASSWORD = TRUE;

GRANT ROLE "USERADMIN" TO USER tips_jr_analyst;

-- for user to see the warehouse, it has to be assigend to that role

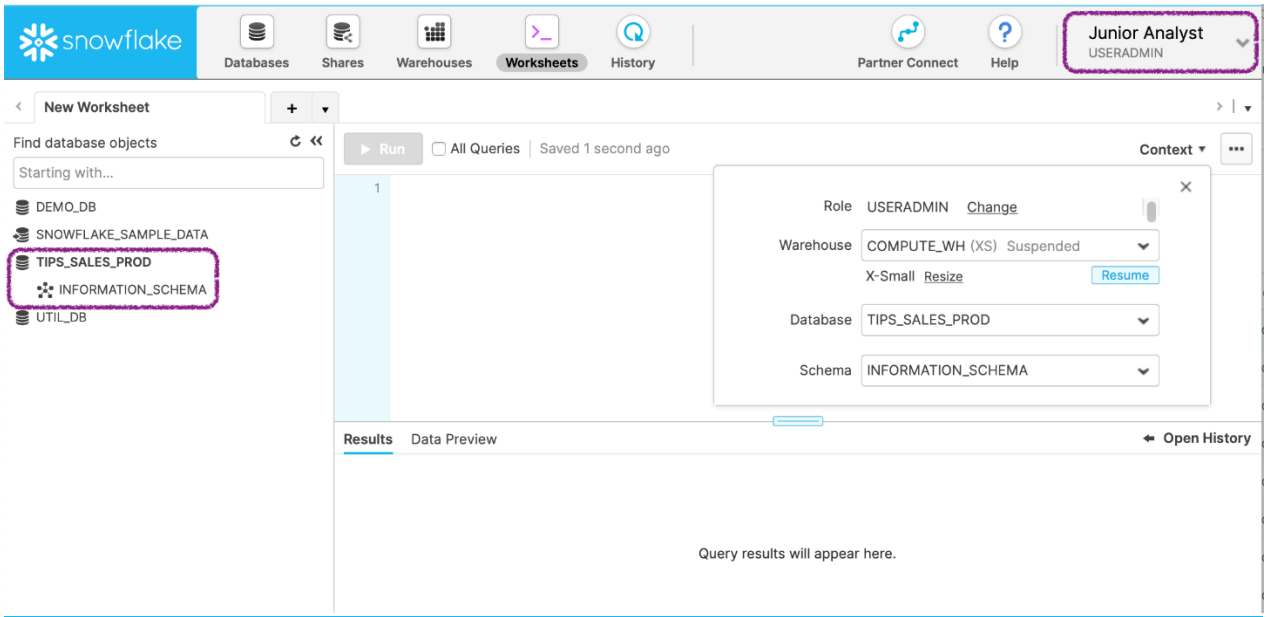
GRANT USAGE, OPERATE, MONITOR ON WAREHOUSE "COMPUTE_WH"

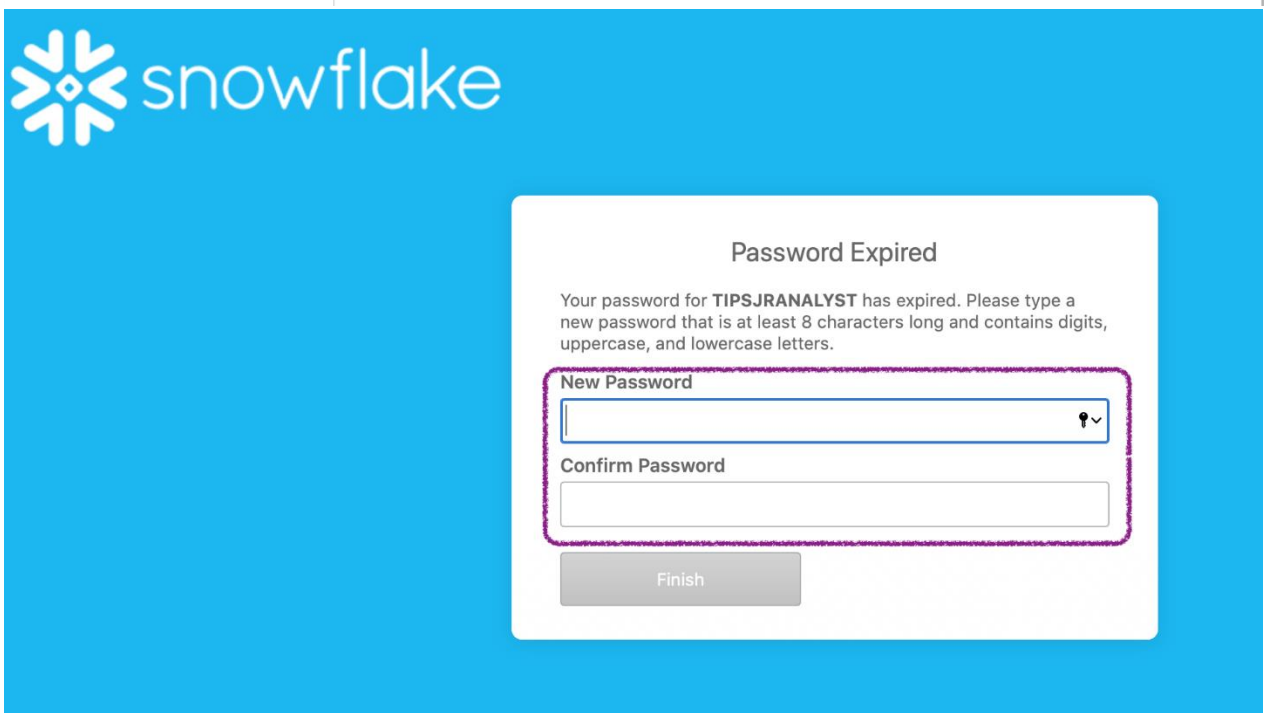
TO ROLE "USERADMIN";

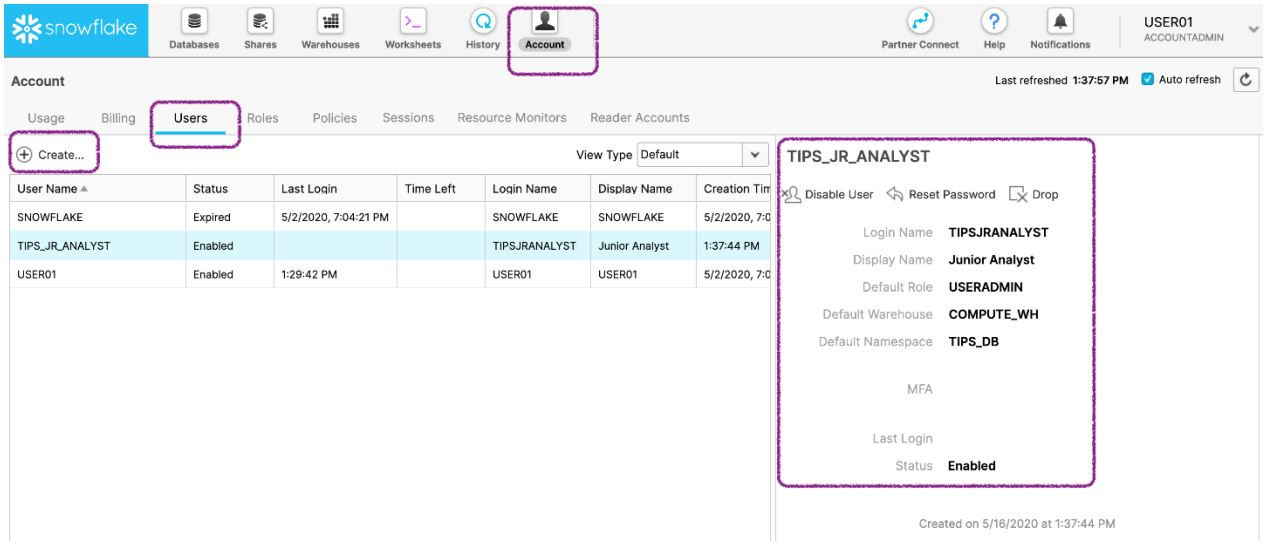
-- grant the database usage

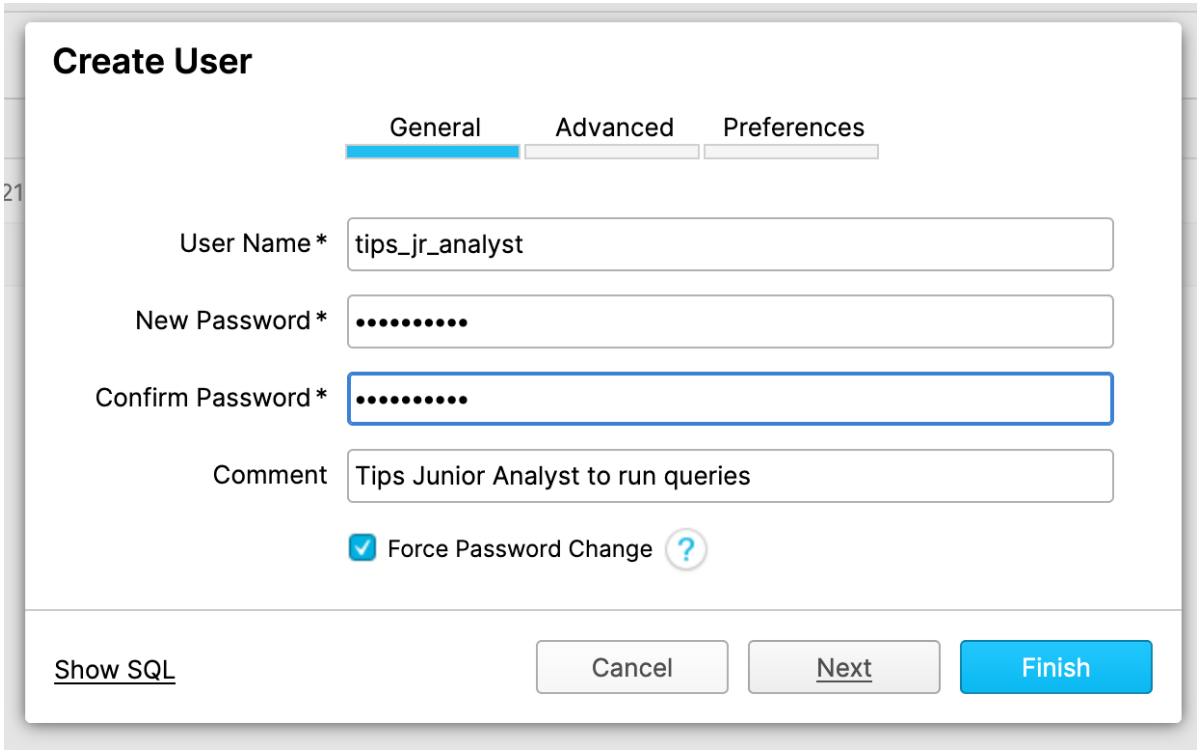
GRANT USAGE, CREATE SCHEMA, MODIFY, MONITOR ON DATABASE "TIPS_SALES_PROD"

TO ROLE "USERADMIN";


snowflake
Databases
Shares
Warehouses
Worksheets
History
Partner Connect
Help
Junior Analyst
USERADMIN
New Worksheet
Find database objects
Starting with...
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
TIPS_SALES_PROD
INFORMATION_SCHEMA
UTIL_DB
Run
All Queries
Saved 1 second ago
Context
Role USERADMIN Change
Warehouse COMPUTE_WH (XS) Suspended
X-Small Resize
Resume
Database TIPS_SALES_PROD
Schema INFORMATION_SCHEMA
Results
Data Preview
Open History
Query results will appear here.


snowflake
Password Expired
Your password for TIPSJRANALYST has expired. Please type a new password that is at least 8 characters long and contains digits, uppercase, and lowercase letters.
New Password
Confirm Password
Finish


snowflake
Databases
Shares
Warehouses
Worksheets
History
Account
Partner Connect
Help
Notifications
USER01
ACCOUNTADMIN
Account
Last refreshed 1:37:57 PM
Auto refresh
Usage
Billing
Users
Roles
Policies
Sessions
Resource Monitors
Reader Accounts
Create...
View Type
Default
User Name
Status
Last Login
Time Left
Login Name
Display Name
Creation Tim
SNOWFLAKE
Expired
5/2/2020, 7:04:21 PM
SNOWFLAKE
SNOWFLAKE
5/2/2020, 7:0
TIPS_JR_ANALYST
Enabled
TIPSJRANALYST
Junior Analyst
1:37:44 PM
USER01
Enabled
1:29:42 PM
USER01
USER01
5/2/2020, 7:0
TIPS_JR_ANALYST
Disable User
Reset Password
Drop
Login Name
TIPSJRANALYST
Display Name
Junior Analyst
Default Role
USERADMIN
Default Warehouse
COMPUTE_WH
Default Namespace
TIPS_DB
MFA
Last Login
Status
Enabled
Created on 5/16/2020 at 1:37:44 PM


Create User
General
Advanced
Preferences
User Name *
tips_jr_analyst
New Password *
Confirm Password *
Comment
Tips Junior Analyst to run queries
Force Password Change
Show SQL
Cancel
Next
Finish

Grant SQL Example

-- Giving a usage grant on a table to public role

GRANT USAGE ON DATABASE "DB_FOR_SECURITY" TO ROLE "PUBLIC" WITH GRANT OPTION;

-- How To: Grant a role access to database objects in a schema

To allow a role to use database objects in a specific schema,

the owner of the database objects (typically a system administrator (SYSADMIN role))

must grant privileges on the database, schema, and objects.

--Grant usage on the database:

GRANT USAGE ON DATABASE <database> TO ROLE <role>;

--Grant usage on the schema:

GRANT USAGE ON SCHEMA <database>.<schema> TO ROLE <role>;

--Grant the ability to query an existing table:

GRANT SELECT ON TABLE <database>.<schema>.<table> TO ROLE <role>;

The following table privileges are supported:

Privilege	**Usage**
SELECT	Execute a SELECT statement on the table.
INSERT	Execute an INSERT command on the table.
UPDATE	Execute an UPDATE command on the table.
TRUNCATE	Execute a TRUNCATE command on the table.
DELETE	Execute a DELETE command on the table.
REFERENCES	Reference the table as the unique/primary key table for a foreign key constraint.
ALL [PRIVILEGES]	Grant all privileges, except OWNERSHIP, on the table.
OWNERSHIP	Grant full control over a table.

Snowflake Name Stages

-- How to show all the stages?

show stages;

list @S3_USDF_EXTERNAL_NAMED_STAGE; --loading all files inside the bucket

list @S3_USDF_EXTERNAL_NAMED_STAGE/load; -- loading specific directory

list @S3_USDF_EXTERNAL_NAMED_STAGE/load/D; -- loading specific directory and file within it (look for * is missing)

-- Note : it is not stage, it is stages, make sure you remember the command.

-- Create a name internal stage (a temporary loading area)

-- This stage will be created within public schema if there is no other schema.

-- you can change the schema and databaes names

```
CREATE OR REPLACE STAGE "USDA_NUTRIENT_STDREF"."PUBLIC".Tips_Internal_Storage
   COMMENT = 'An internal staging area';
```

-- The grant query to be documented?

-- Create a named external stage (a temporary loading area) with AWS S3 (Simple Storage Services)

-- This stage will be created within public schema if there is no other schema.

-- you can change the schema and databaes names

```
CREATE OR REPLACE STAGE "USDA_NUTRIENT_STDREF"."PUBLIC".S3_Tips_External_Named_Stage
   URL = 's3://my-sample-bucket'
   CREDENTIALS = (AWS_KEY_ID = 'myuser' AWS_SECRET_KEY = '********')
   ENCRYPTION = (MASTER_KEY = '*********')
   COMMENT = 'S3_Tips_External_Named_Stage';
```

-- Creating a temporary storage

-- Not documentation is found how it is different from normal stage and what is the purpose

-- This is to be explroed ???

```
CREATE OR REPLACE TEMPORARY STAGE
"USDA_NUTRIENT_STDREF"."PUBLIC".S3_TEMPO_EXTERNAL_NAMED_STAGE
   URL = 's3://my-sample-bucket'
   CREDENTIALS = (AWS_KEY_ID = 'myuser' AWS_SECRET_KEY = '********')
   COMMENT = 'Checking what is this temporary stage is';
```

-- Alter Stage (Changing Paramters)

```
ALTER STAGE "USDA_NUTRIENT_STDREF"."PUBLIC"."S3_USDF_EXTERNAL_NAMED_STAGE"
   SET URL = 's3://on-demand-files'
   COMMENT = 'S3_USDF_EXTERNAL_NAMED_STAGS';
```

-- Alter rename does not work, looks like it is not supported as UI also does not allow.

```
ALTER STAGE IF EXISTS
"USDA_NUTRIENT_STDREF"."PUBLIC".S3_TEMPO_EXTERNAL_NAMED_STAGE
RENAME "USDA_NUTRIENT_STDREF"."PUBLIC".S3_TEMPO_EXTERNAL_NAMED_STAGE_RENAMED
```

-- (this will end with error : SQL compilation error: syntax error line 2 at position 7 unexpected '"USDA_NUTRIENT_STDREF"'.)

-- Even unset comment does not seems to be working

-- (error : Unsupported feature 'UNSET'.)

-- Note-01: if you input incorrect AWS keys, Snowflake will throw error like this

--Unable to create stage "S3_TIPS_EXTERNAL_NAMED_STAGE".

--The provided master key has invalid length. It must be either 128 bits, 192 bits, or 256 bits long.

-- Note-02: If the bucket(s3) is publicly available, then you don't need to specify the key at all.

-- Creating External Named Stages with Azure

```
CREATE OR REPLACE STAGE
"USDA_NUTRIENT_STDREF"."PUBLIC".Azure_Tips_External_Named_Stage
   URL = 'azure://azurebucket'
   CREDENTIALS = (AZURE_SAS_TOKEN = '*********************')
   ENCRYPTION = (TYPE = 'AZURE_CSE' MASTER_KEY =
'**************************************')
   COMMENT = 'Azure_Tips_External_Named_Stage';
```

-- How to create a stage by cloing it

-- Creating a clone works for external stage but does not work for internal stages

```
CREATE STAGE "USDA_NUTRIENT_STDREF"."PUBLIC".Cloned_s3_tips
   CLONE "USDA_NUTRIENT_STDREF"."PUBLIC"."S3_TIPS_EXTERNAL_NAMED_STAGE"
   COMMENT = 'Cloning an s3 named stage';
```

-- Note-01 : If you try to create a clone using internal stages, it will throw error

Unable to create stage "CLONE_INTERNAL_STAGE".

Unsupported feature 'Cloning internal and temporary stages'.

-- Note-02 : Grants are also not copied when you clone

Different kind of error may appear if you are not having sufficient privileges

so make sure you check the role using the role manu and then run the queries or operations.

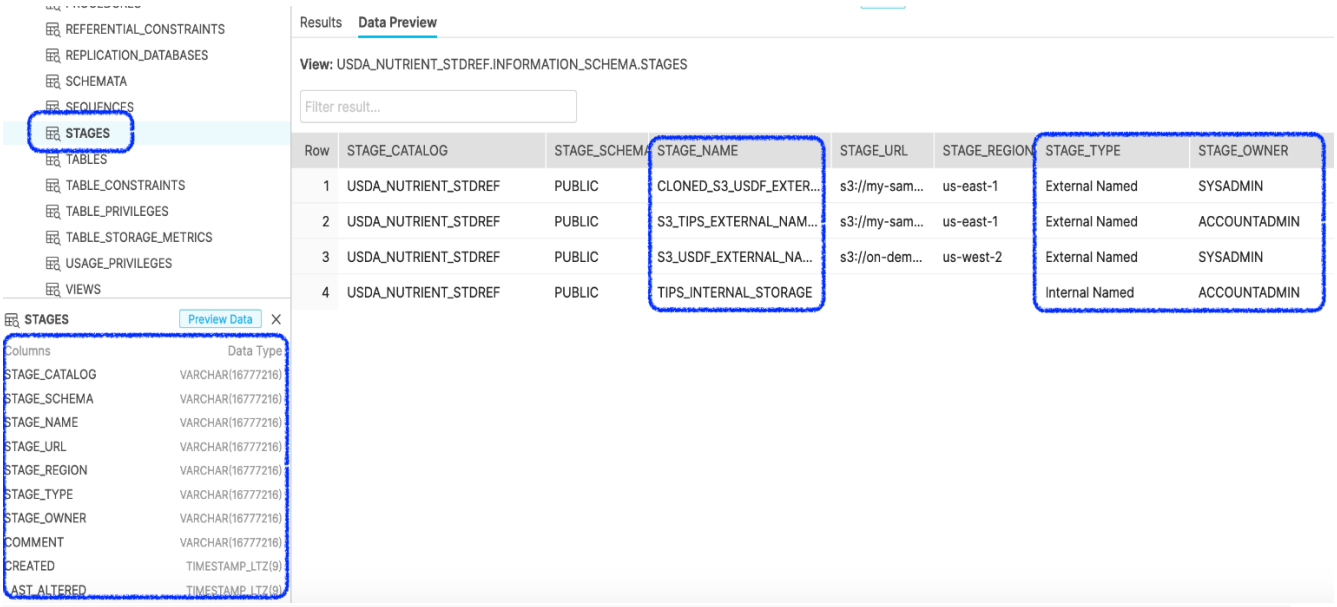
Check for the ICON called "Owner" on WebUI before operating on any staged object

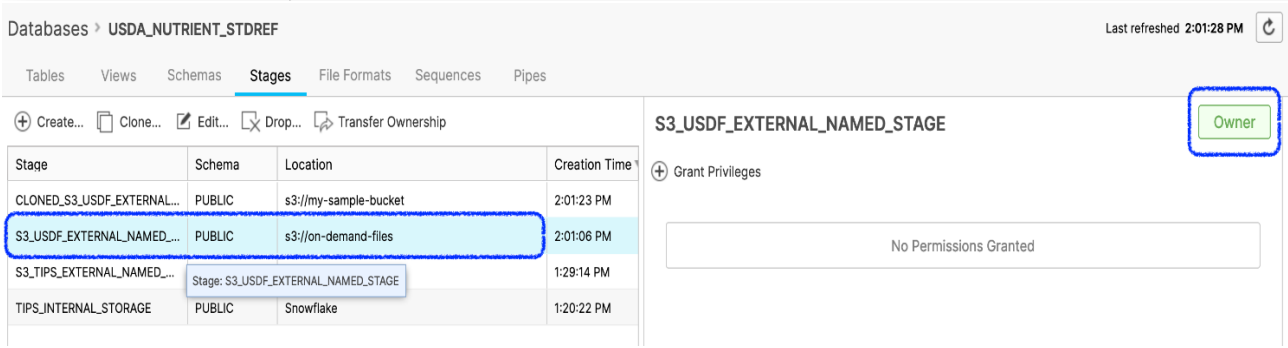
Unable to modify stage "S3_TIPS_EXTERNAL_NAMED_STAGE".

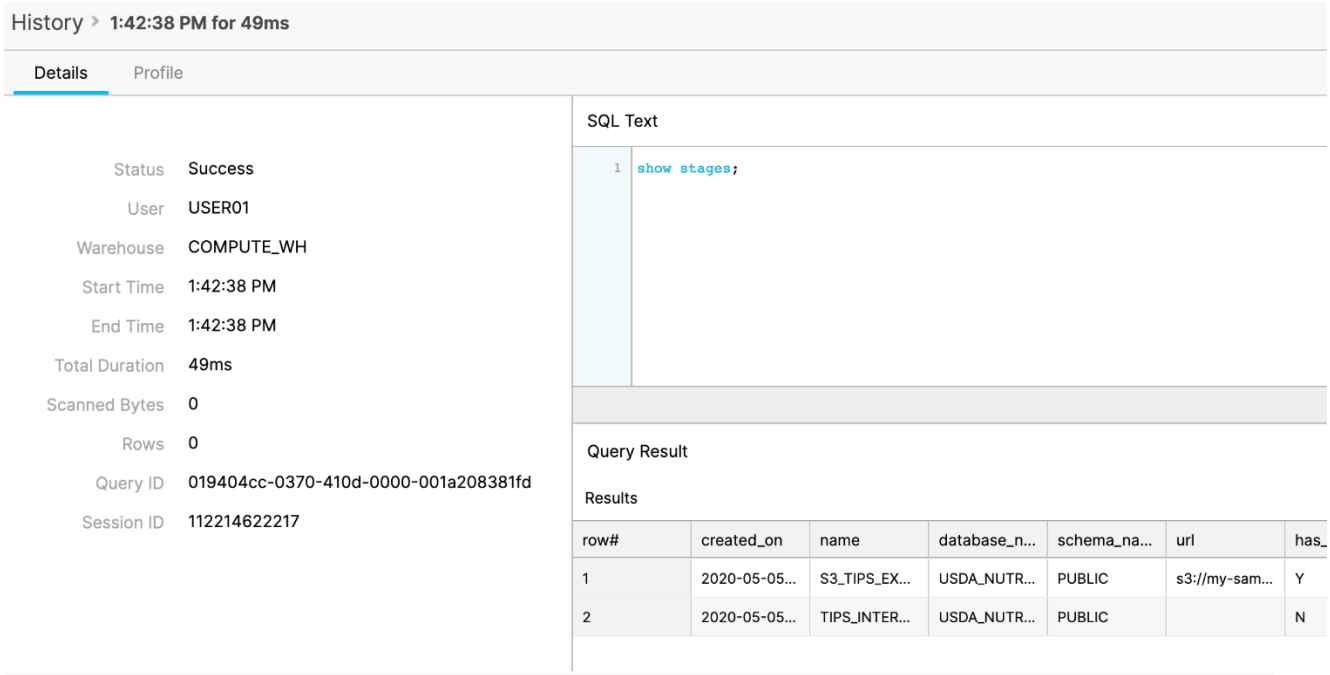
SQL access control error: Insufficient privileges to operate on stage 'S3_TIPS_EXTERNAL_NAMED_STAGE'

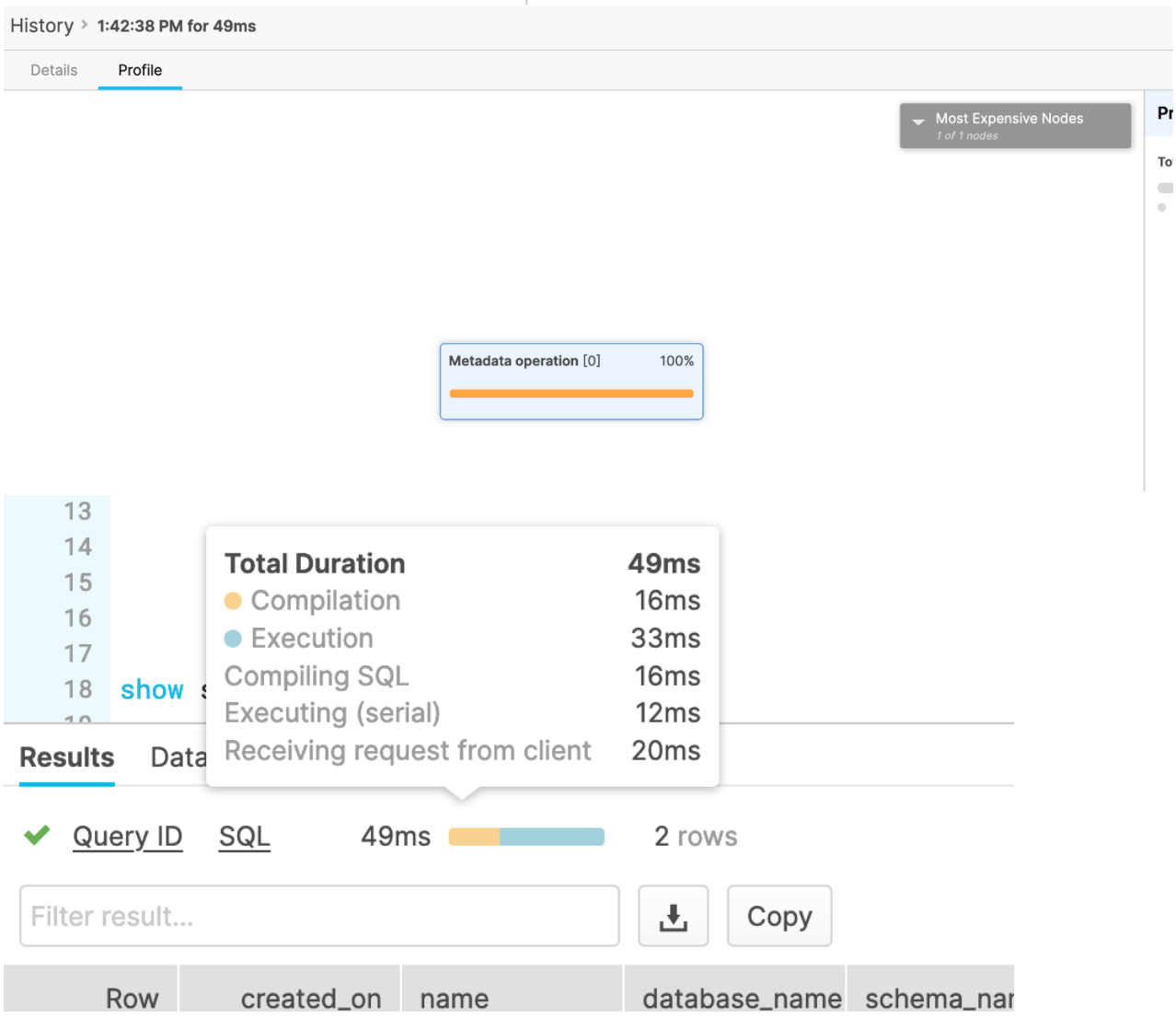
Copy

ption_	owner	comment	region	type	cloud	notification_cha
	SYSADMIN	CLONED_S3...	us-east-1	EXTERNAL	AWS	NULL
	SYSADMIN	Checking w...	us-west-2	EXTERNAL TEMPORARY	AWS	NULL
	ACCOUNTA...	S3_Tips_Ext...	us-east-1	EXTERNAL	AWS	NULL
	SYSADMIN	S3_USDF_EX...	us-west-2	EXTERNAL	AWS	NULL
	ACCOUNTA...	An internal s...	NULL	INTERNAL	NULL	NULL






History
1:42:38 PM for 49ms
Details
Profile
Status Success
User USER01
Warehouse COMPUTE_WH
Start Time 1:42:38 PM
End Time 1:42:38 PM
Total Duration 49ms
Scanned Bytes 0
Rows 0
Query ID 019404cc-0370-410d-0000-001a208381fd
Session ID 112214622217
SQL Text
1 show stages;
Query Result
Results
row# created_on name database_n... schema_na... url has_
1 2020-05-05... S3_TIPS_EX... USDA_NUTR... PUBLIC s3://my-sam... Y
2 2020-05-05... TIPS_INTER... USDA_NUTR... PUBLIC N


History
1:42:38 PM for 49ms
Details
Profile
Most Expensive Nodes
1 of 1 nodes
Metadata operation [0] 100%
13
14
15
16
17
18 show
Total Duration 49ms
Compilation 16ms
Execution 33ms
Compiling SQL 16ms
Executing (serial) 12ms
Receiving request from client 20ms
Results
Data
Query ID SQL 49ms 2 rows
Filter result...
Copy
Row created_on name database_name schema_nar

```
18 show stages;
```

Results | Data Preview

Query ID | SQL | 49ms | 2 rows

Filter result... | Copy

Row	created_on	name	database_name	schema_name	url	has_credentials	has_encryption_	owner	comment	region
1	2020-05-05...	S3_TIPS_EX...	USDA_NUTR...	PUBLIC	s3://my-sam...	Y	N	ACCOUNTA...	S3_Tips_Ext...	us-east-1
2	2020-05-05...	TIPS_INTER...	USDA_NUTR...	PUBLIC		N	N	ACCOUNTA...	An internal s...	NULL

Clone Stage

Name * Cloned_s3_tips

Source PUBLIC.S3_TIPS_EXTERNAL_NAMED_STAGE

Comment Cloning an s3 named stage

Show SQL | Cancel | Finish

Create Stage

Staged files will be stored in the specified Azure location

Name * Azure_Tips_External_Named_Stage

Schema Name PUBLIC

URL * azure://azurebucket

Azure SAS Token ••••••••••••••••••••

Encryption Master Key ••

Comment Azure_Tips_External_Named_Stage

Show SQL | Cancel | Back | Finish

Create Stage

Staged files will be stored in the specified S3 location

Name * S3_Tips_External_Named_Stage

Schema Name PUBLIC

URL * s3://my-sample-bucket

AWS Key ID user01

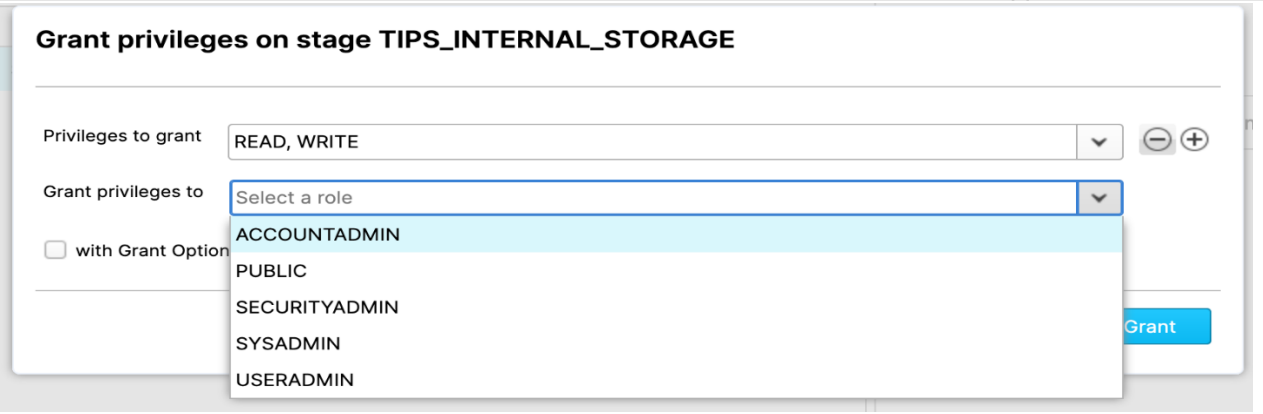
AWS Secret Key ••••••••

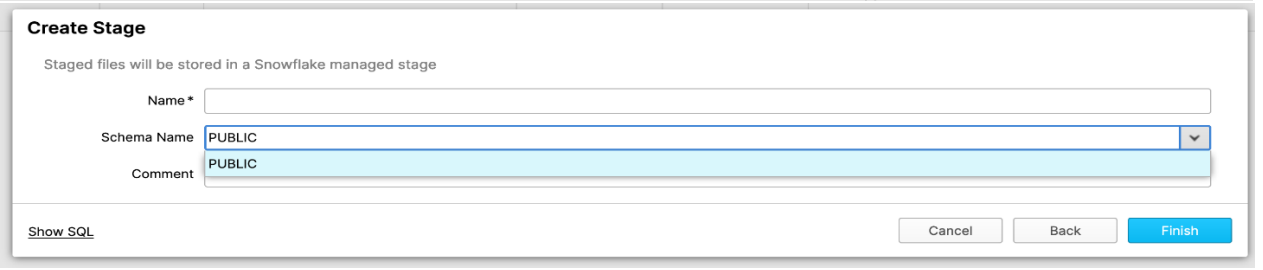
Encryption Master Key •••••••••

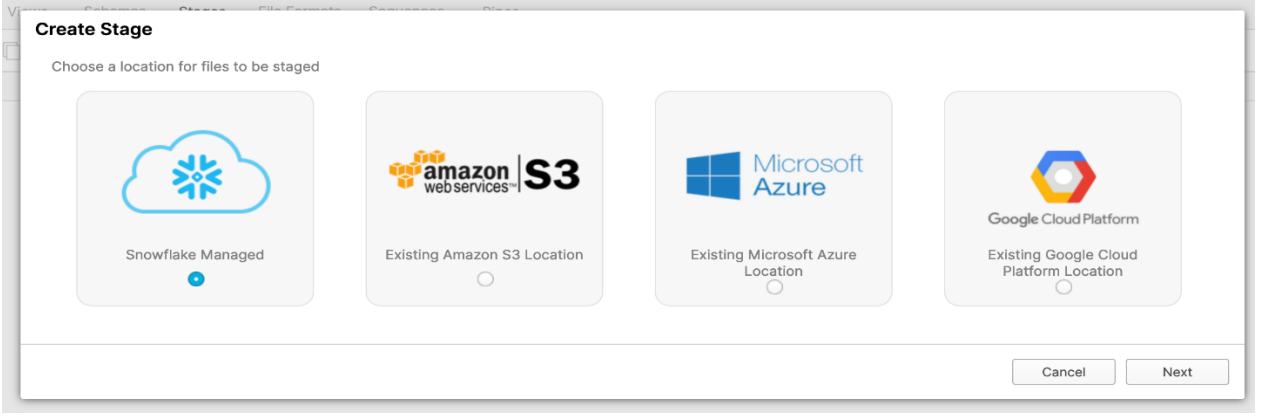
Comment S3_Tips_External_Named_Stage

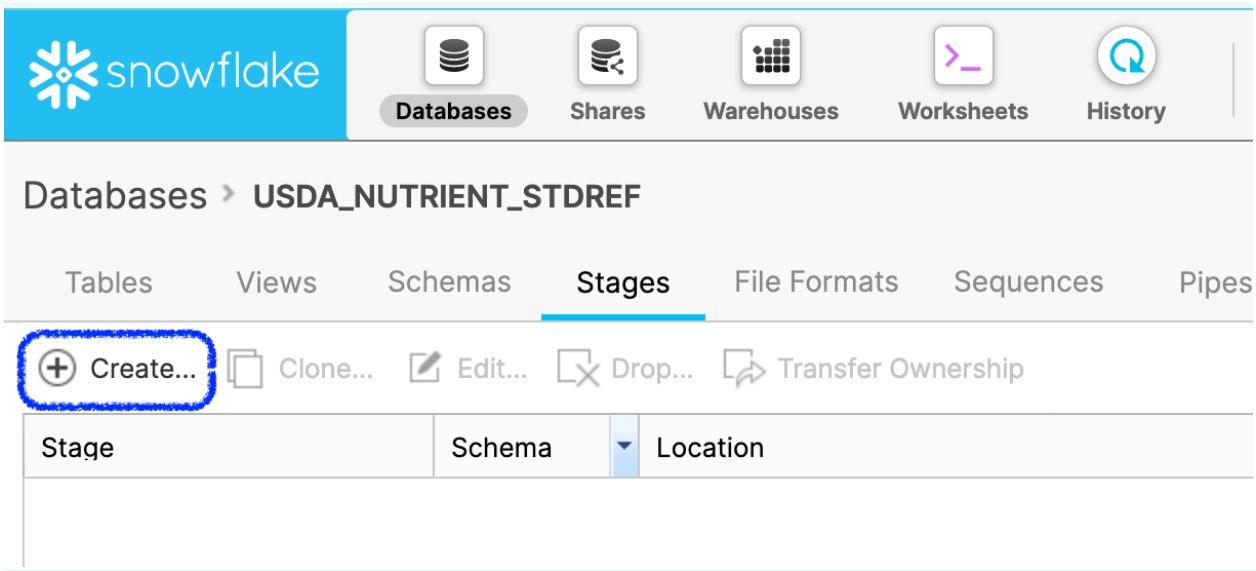
Show SQL | Cancel | Back | Finish

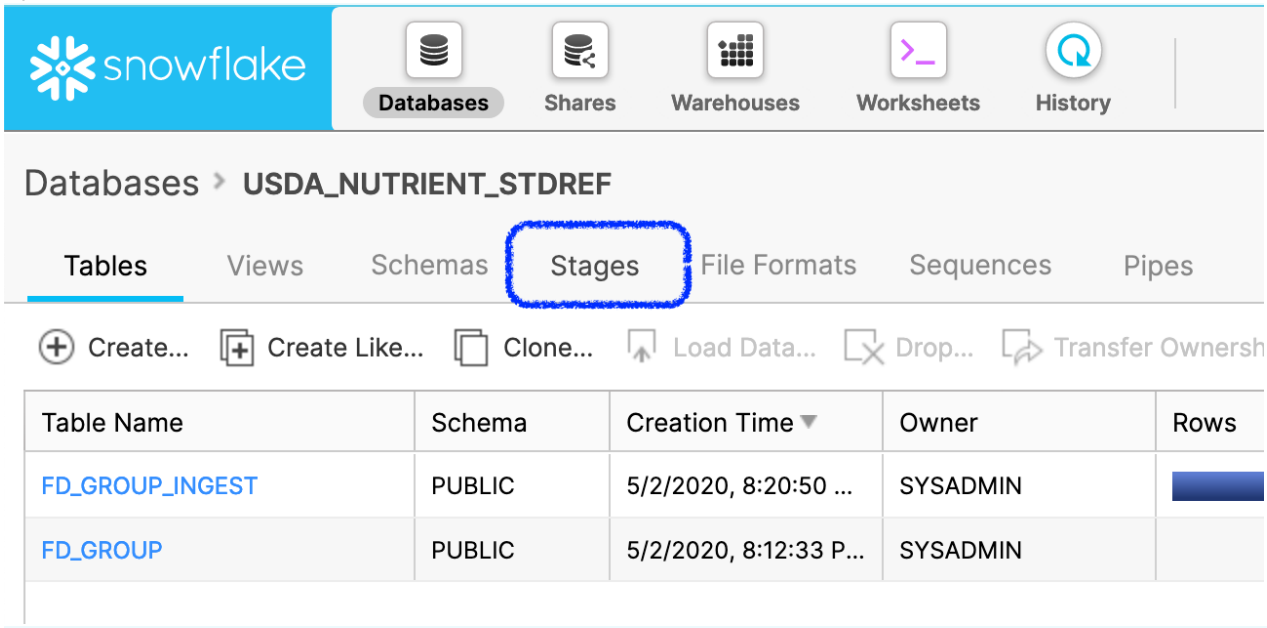

Edit privileges on stage TIPS_INTERNAL_STORAGE
Privileges to grant
READ, WRITE
Grant privileges to
PUBLIC
with Grant Option
Revoke All
Cancel
Grant

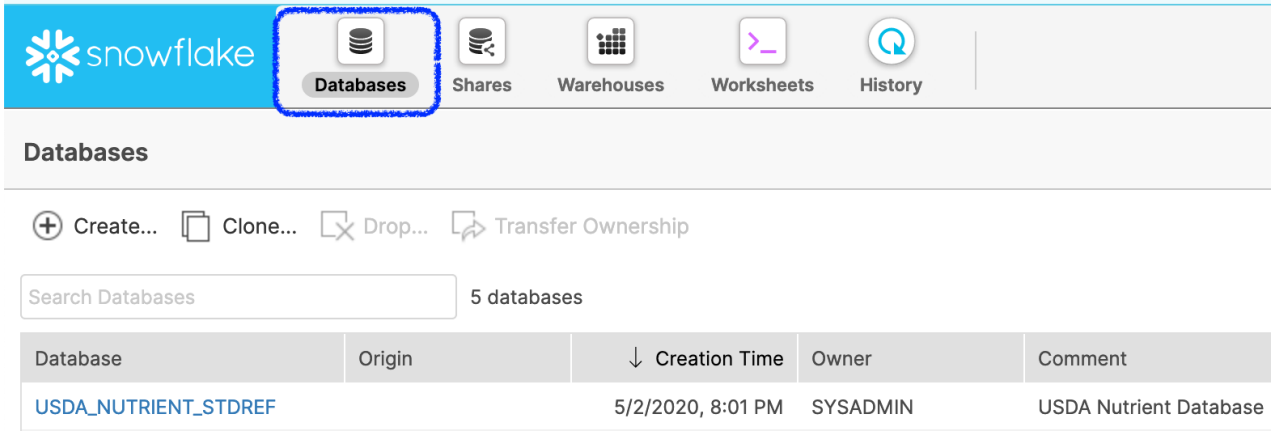

Grant privileges on stage TIPS_INTERNAL_STORAGE
Privileges to grant
READ, WRITE
Grant privileges to
Select a role
with Grant Option
ACCOUNTADMIN
PUBLIC
SECURITYADMIN
SYSADMIN
USERADMIN
Grant


Create Stage
Staged files will be stored in a Snowflake managed stage
Name *
Schema Name
PUBLIC
Comment
PUBLIC
Show SQL
Cancel
Back
Finish


Create Stage
Choose a location for files to be staged
Snowflake Managed
Existing Amazon S3 Location
Existing Microsoft Azure Location
Existing Google Cloud Platform Location
Cancel
Next


snowflake
Databases
Shares
Warehouses
Worksheets
History
Databases › USDA_NUTRIENT_STDREF
Tables
Views
Schemas
Stages
File Formats
Sequences
Pipes
Create...
Clone...
Edit...
Drop...
Transfer Ownership
Stage
Schema
Location


snowflake
Databases
Shares
Warehouses
Worksheets
History
Databases › USDA_NUTRIENT_STDREF
Tables
Views
Schemas
Stages
File Formats
Sequences
Pipes
Create...
Create Like...
Clone...
Load Data...
Drop...
Transfer Ownersh
Table Name
Schema
Creation Time
Owner
Rows
FD_GROUP_INGEST
PUBLIC
5/2/2020, 8:20:50 ...
SYSADMIN
FD_GROUP
PUBLIC
5/2/2020, 8:12:33 P...
SYSADMIN


snowflake
Databases
Shares
Warehouses
Worksheets
History
Databases
Create...
Clone...
Drop...
Transfer Ownership
Search Databases
5 databases
Database
Origin
Creation Time
Owner
Comment
USDA_NUTRIENT_STDREF
5/2/2020, 8:01 PM
SYSADMIN
USDA Nutrient Database

Role

Important Notes

- For any role to function, we need at least one user assigned to it

- SYSADMIN role by default cannot create new roles or users. role creation operation will fail

```
create role tips_jr_dba;
grant role tips_jr_dba to user tips_dba_user02;
use role tips_jr_dba;

-- Now we have to give usage access to this new role
use role accountadmin;
grant usage on database citibike to role tips_jr_dba;
grant usage on database weather to role tips_jr_dba;
```

Hive

Hive Function Meta commands

- SHOW FUNCTIONS– lists Hive functions and operators
- DESCRIBE FUNCTION [function name]– displays short description of the function
- DESCRIBE FUNCTION EXTENDED [function name]– access extended description of the function

Types of Hive Functions

- UDF– is a function that takes one or more columns from a row as argument and returns a single value or object. Eg: concat(col1, col2)
- UDAF- aggregates column values in multiple rows and returns a single value. Eg: sum(c1)
- UDTF— takes zero or more inputs and and produces multiple columns or rows of output. Eg: explode()
- Macros— a function that users other Hive functions.

Mathematical Functions

Return Type	Name (Signature)	Description
BIGINT	round(double a)	Returns the rounded BIGINT value of the double
DOUBLE	round(double a, int d)	Returns the double rounded to d decimal places
BIGINT	floor(double a)	Returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a), ceiling(double a)	Returns the minimum BIGINT value that is equal or greater than the double
double	rand(), rand(int seed)	Returns a random number (that changes from row to row) that is distributed uniformly from 0 to 1. Specifiying the seed will make sure the generated random number sequence is deterministic.
double	exp(double a)	Returns ea where e is the base of the natural logarithm
double	ln(double a)	Returns the natural logarithm of the argument
double	log10(double a)	Returns the base-10 logarithm of the argument
double	log2(double a)	Returns the base-2 logarithm of the argument
double	log(double base, double a)	Return the base "base" logarithm of the argument
double	pow(double a, double p), power(double a, double p)	Return ap
double	sqrt(double a)	Returns the square root of a
string	bin(BIGINT a)	Returns the number in binary format
string	hex(BIGINT a) hex(string a)	If the argument is an int, hex returns the number as a string in hex format. Otherwise if the number is a string, it converts each character into its hex representation and returns the resulting string.
string	unhex(string a)	Inverse of hex. Interprets each pair of characters as a hexidecimal number and converts to the character represented by the number.
string	conv(BIGINT num, int from_base, int to_base), conv(STRING num, int from_base, int to_base)	Converts a number from a given base to another
double	abs(double a)	Returns the absolute value
int double	pmod(int a, int b) pmod(double a, double b)	Returns the positive value of a mod b
double	sin(double a)	Returns the sine of a (a is in radians)
double	asin(double a)	Returns the arc sin of x if -1<=a<=1 or null otherwise
double	cos(double a)	Returns the cosine of a (a is in radians)
double	acos(double a)	Returns the arc cosine of x if -1<=a<=1 or null otherwise
double	tan(double a)	Returns the tangent of a (a is in radians)
double	atan(double a)	Returns the arctangent of a
double	degrees(double a)	Converts value of a from radians to degrees
double	radians(double a)	Converts value of a from degrees to radians
int double	positive(int a), positive(double a)	Returns a
int double	negative(int a), negative(double a)	Returns -a
float	sign(double a)	Returns the sign of a as '1.0' or '-1.0'
double	e()	Returns the value of e
double	pi()	Returns the value of pi

Date Functions

Return Type	Name (Signature)	Description
string	from_unixtime(bigint unixtime[, string format])	Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
bigint	unix_timestamp()	Gets current time stamp using the default time zone.
bigint	unix_timestamp(string date)	Converts time string in format yyyy-MM-dd HH:mm:ss to Unix time stamp, return 0 if fail: unix_timestamp ('2009-03-20 11:30:01') = 1237573801
bigint	unix_timestamp(string date, string pattern)	Convert time string with given pattern to Unix time stamp, return 0 if fail: unix_timestamp('2009-03-20', 'yyyy-MM-dd') = 1237532400
string	to_date(string timestamp)	Returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	Returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	Returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month ("1970-11-01") = 11
int	day(string date) dayofmonth(date)	Return the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
int	hour(string date)	Returns the hour of the timestamp: hour('2009-07-30 12:58:59') = 12, hour('12:58:59') = 12
int	minute(string date)	Returns the minute of the timestamp
int	second(string date)	Returns the second of the timestamp
int	weekofyear(string date)	Return the week number of a timestamp string: weekofyear("1970-11-01 00:00:00") = 44, weekofyear ("1970-11-01") = 44
int	datediff(string enddate, string startdate)	Return the number of days from startdate to enddate: datediff('2009-03-01', '2009-02-27') = 2
string	date_add(string startdate, int days)	Add a number of days to startdate: date_add('2008-12-31', 1) = '2009-01-01'
string	date_sub(string startdate, int days)	Subtract a number of days to startdate: date_sub('2008-12-31', 1) = '2008-12-30'
timestamp	from_utc_timestamp(timestamp, string timezone)	Assumes given timestamp ist UTC and converts to given timezone (as of Hive 0.8.0)
timestamp	to_utc_timestamp(timestamp, string timezone)	Assumes given timestamp is in given timezone and converts to UTC (as of Hive 0.8.0)

Hive

There is no direct way to check the version. Run the below command to check the version.

hive> set system:sun.java.command;

Move a file to hdfs first

ls ~/labs/map-reduce/hive-lab/free-txt-files/india.txt

hdfs dfs -mkdir labs

hdfs dfs -put ~/labs/map-reduce/hive-lab/free-txt-files/india.txt labs/

hdfs dfs -ls labs/india.txt

A simple hive code - word count example

1) connect to hive shell

2) create database labs;

3) create tables docs (line

CREATE TABLE docs (line STRING);

```
LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE docs;


CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\s')) AS word FROM docs) w
GROUP BY word
ORDER BY word;
```

Hive is not a standalone tool and relies on various components for storing and querying data.

Within the Hadoop ecosystem, Hive is considered a client data access tool. Data access requires a compute, storage, management, as well as a security framework

Topic 1: Primitive Data Types

Type	Size	Example
.....	.....	.....
TINYINT	1 byte signed integer.	20
SMALLINT	2 byte signed integer.	20
INT	4 byte signed integer.	20
BIGINT	8 byte signed integer.	20
BOOLEAN	Boolean true or false.	TRUE
FLOAT	Single precision floating point.	3.14159
DOUBLE	Double precision floating point.	3.14159
STRING	Sequence of characters.	'Now is the time'
BINARY	Array of bytes. See discussion below	

It’s useful to remember that each of these types is implemented in Java.

So the particular behavior details will be exactly what you would expect from the corresponding Java types.

For example, STRING is implemented by the Java String, FLOAT is implemented by Java float, etc.

Values of the new TIMESTAMP type can be integers, which are interpreted as seconds since the Unix epoch time (Midnight, January 1, 1970), floats, which are interpreted as seconds

since the epoch time with nanosecond resolution (up to 9 decimal places), and strings, which are interpreted according to the JDBC date string format convention,

YYYY-MM-DD hh:mm:ss.fffffffff.

Collection Data Types

STRUCT Analogous to a C struct or an “object.”
struct('John', 'Doe')

Fields can be accessed using the “dot” notation.

For example, if a column name is of type STRUCT

{first STRING; last STRING}, then the

first name field can be referenced using name.first.

MAP A collection of key-value tuples,
map('first', 'John', 'last', 'Doe')

where the fields are accessed using array notation (e.g., ['key']).

For example, if a column name is of type MAP with key→value pairs

'first'→'John' and 'last'→'Doe', then the last name can be

referenced using name['last'].

ARRAY Ordered sequences of the same type that are indexable using
array('John', 'Doe')

zero-based integers. For example, if a column name is of type

ARRAY of strings with the value ['John', 'Doe'], then

the second element can be referenced using name[1].

Most relational databases don’t support such collection types, because using them tends to break normal form.

For example, in traditional data models, structs might be captured in separate tables, with foreign key relations between the tables, as appropriate.

A practical problem with breaking normal form is the greater risk of data duplication, leading to unnecessary disk space consumption and potential data inconsistencies, as duplicate copies can grow out of sync as changes are made

Here is a table declaration that demonstrates how to use these types, an employees table in a fictitious Human Resources application:

```
CREATE TABLE employees (
name STRING,
salary FLOAT,
subordinates ARRAY<STRING>,
deductions MAP<STRING, FLOAT>,
address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>);
```

Hive's default record and field delimiters

---

1) \n => For text files, each line is a record, so the line feed character separates records.

2) ^A ("control" A) Separates all fields (columns). Written using the octal code \001 when explicitly specified in CREATE TABLE statements.

2) ^B Separate the elements in an ARRAY or STRUCT, or the key-value pairs in a MAP. Written using the octal code \002 when explicitly specified in CREATE TABLE statements.

3) ^C Separate the key from the corresponding value in MAP key-value pairs. Written using the octal code \003 when explicitly specified in CREATE TABLE statements.

```
CREATE TABLE employees (
    name STRING,
    salary FLOAT,
    subordinates ARRAY<STRING>,
    deductions MAP<STRING, FLOAT>,
    address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
    )
ROW FORMAT DELIMITED
```

FIELDS TERMINATED BY '\001'

COLLECTION ITEMS TERMINATED BY '\002'

MAP KEYS TERMINATED BY '\003'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

Note :

The ROW FORMAT DELIMITED sequence of keywords must appear before any of the other clauses.

The character \001 is the octal code for ^A. The clause ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001' means that Hive will use the ^A character to separate fields.

Similarly, the character \002 is the octal code for ^B. The clause ROW FORMAT DELIMITED COLLECTION ITEMS TERMINATED BY '\002' means that Hive will use the ^B character to separate collection items.

Finally, the character \003 is the octal code for ^C. The clause ROW FORMAT DELIMITED MAP KEYS TERMINATED BY '\003' means that Hive will use the ^C character to separate map keys from values.

Database in Hive & Data Definition

---

1) HiveQL is the Hive query language.

2) Like all SQL dialects in widespread use, it doesn't fully conform to any particular revision of the ANSI SQL standard.

3) It is perhaps closest to MySQL's dialect.

4) Hive offers no support for rowlevel inserts, updates, and deletes. Hive doesn't support transactions.

5) If you don't specify a database, the default database is used.

6) The Hive concept of a database is essentially just a catalog or namespace of tables

Lets Practice

---

1) CREATE DATABASE sales;

2) CREATE DATABASE IF NOT EXISTS sales;

3) SHOW DATABASES;

4) SHOW DATABASES LIKE 's.*';

5) hive> CREATE DATABASE IF NOT EXISTS sales

> COMMENT 'Holds all financial tables';

6) DESCRIBE DATABASE sales;

7) you can associate key-value properties with the database, although their only function currently is to provide a way of adding information to the output of DESCRIBE DATABASE EXTENDED <database>:

hive> CREATE DATABASE sales7

> WITH DBPROPERTIES ('creator' = 'Mark Moneybags', 'date' = '2012-01-02');

8) DESCRIBE DATABASE EXTENDED sales7;

9) user sales - is the command to use the database, however there is no command to see which database is being used.

10) Setting a property to print the current database as part of the prompt

hive> set hive.cli.print.current.db=true;

11) DROP DATABASE IF EXISTS sales;

The IF EXISTS is optional and suppresses warnings if financials doesn't exist.

By default, Hive won't permit you to drop a database if it contains tables.

You can either drop the tables first or append the CASCADE keyword to the command, which will cause the Hive to drop the tables in the database first:

hive> DROP DATABASE IF EXISTS financials CASCADE;

When a database is dropped, its directory is also deleted

12) Alter Database

12.1) You can set key-value pairs in the DBPROPERTIES associated with a database using the ALTER DATABASE command.

```
hive> ALTER DATABASE sales SET DBPROPERTIES ('edited-by' = 'Labs');
```

Imp **** There is no way to delete or “unset” a DBPROPERTY.

13) Creating Tables

```
CREATE TABLE IF NOT EXISTS sales.employees (
    name STRING COMMENT 'Employee name',
    salary FLOAT COMMENT 'Employee salary',
    subordinates ARRAY<STRING> COMMENT 'Names of subordinates',
    deductions MAP<STRING, FLOAT>
    COMMENT 'Keys are deductions names, values are percentages',
    address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
    COMMENT 'Home address')
    COMMENT 'Description of the table'
    TBLPROPERTIES ('creator'='me', 'created_at'='2012-01-02 10:00:00', ...)
    LOCATION '/user/hive/warehouse/mydb.db/employees';
```

Imp Note

---

* Hive automatically adds two table properties: last_modified_by holds the username of the last user to modify the table, and last_modified_time holds the epoch time in seconds of that modification.

* show tblproperties employees

14) You can also copy the schema (but not the data) of an existing table:

```
CREATE TABLE IF NOT EXISTS sales.employees2
LIKE sales.employees;
```

15) see the tables

show tables;

show tables emp.*;

show tables in sales;

16) Describe the table

DESCRIBE EXTENDED sales.employees;

DESCRIBE FORMATTED sales.employees;

17) If you only want to see the schema for a particular column, append the column to the table name

describe sales.employee

Managed Tables in Hive

The tables we have created so far are called managed tables or sometimes called internal tables, because Hive controls the lifecycle of their data (more or less). As we've seen,

Hive stores the data for these tables in a subdirectory under the directory defined by hive.metastore.warehouse.dir (e.g., /user/hive/warehouse)

run this command and see hdfs dfs -ls /user/hive/warehouse

When we drop a managed table, Hive deletes the data in the table.

External Table

Suppose we are analyzing data from the stock markets. Periodically, we ingest the data for NASDAQ and the NYSE.

https://www.nasdaq.com/symbol/csv/historical

CREATE EXTERNAL TABLE IF NOT EXISTS stocks (

date STRING,

close FLOAT,

volume INT,

```
open FLOAT,
high FLOAT,
low FLOAT)
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    LOCATION '/landing/stocks';
```

** The EXTERNAL keyword tells Hive this table is external and the LOCATION … clause is required to tell Hive where it's located.

** Because it's external, Hive does not assume it owns the data. Therefore, dropping the table does not delete the data, although the metadata for the table will be deleted.

Copy schema but change the data set

```
CREATE EXTERNAL TABLE IF NOT EXISTS sales.employees3
LIKE sales.employees
LOCATION '/path/to/data';
```

** If you omit the EXTERNAL keyword and the original table is external, the new table will also be external

** If you omit EXTERNAL and the original table is managed, the new table will also be managed.

Alter Tables Statements

Most table properties can be altered with ALTER TABLE statements, which change metadata about the table but not the data itself.

Renaming a Table : ALTER TABLE employee RENAME TO xemployee;

Changing Column :

```
ALTER TABLE employee
CHANGE COLUMN salary total_salary INT
COMMENT 'changing it'
AFTER deduction;
```

Adding Column :

```
ALTER TABLE log_messages ADD COLUMNS (
app_name STRING COMMENT 'Application name',
session_id LONG COMMENT 'The current session id');
```

Deleting or Replacing Columns :

```
ALTER TABLE log_messages REPLACE COLUMNS (
hours_mins_secs INT COMMENT 'hour, minute, seconds from timestamp',
severity STRING COMMENT 'The message severity'
message STRING COMMENT 'The rest of the message');
ALTER TABLE log_messages ADD IF NOT EXISTS
PARTITION (year = 2011, month = 1, day = 1) LOCATION '/logs/2011/01/01'
PARTITION (year = 2011, month = 1, day = 2) LOCATION '/logs/2011/01/02'
PARTITION (year = 2011, month = 1, day = 3) LOCATION '/logs/2011/01/03'
```

Alter Storage Properties

```
ALTER TABLE log_messages
PARTITION(year = 2012, month = 1, day = 1)
SET FILEFORMAT SEQUENCEFILE;
```

Inserting Data Into Hive Table

Partitioned, Managed Tables

It’s used for distributing load horizontally, moving data physically closer to its most frequent users, and other purposes

```
CREATE TABLE employees (
        name STRING,
        salary FLOAT,
        subordinates ARRAY<STRING>,
        deductions MAP<STRING, FLOAT>,
        address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
        )
PARTITIONED BY (country STRING, state STRING);
```

Partitioning tables changes how Hive structures the data storage. If we create this table in the mydb database, there will still be an employees directory for the table:

```
...
.../employees/country=CA/state=AB
.../employees/country=CA/state=BC
...
.../employees/country=US/state=AL
.../employees/country=US/state=AK
...
```

You can see the partitions that exist with the SHOW PARTITIONS command:

```
hive> SHOW PARTITIONS employees;
```

Key Configuration

---

1) The database directory is created under a top-level directory specified by the property hive.metastore.warehouse.dir

2) You can override this default location for the new directory as shown in this example CREATE DATABASE sales LOCATION '/my/preferred/directory';

Complex Structure

-- define some default properties

set hive.cli.print.header=true;

set hive.cli.print.header=true;

-- Complex Data Type Example

```
drop table if exists employee_internal;
CREATE TABLE employee_internal
(
name string,
work_place ARRAY<string>,
gender_age STRUCT<gender:string,age:smallint>,
skills_score MAP<string,smallint>,
depart_title MAP<string,ARRAY<string>>
)
COMMENT 'This is an internal table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

describe formatted employee_internal;
```

Sample Data File : employee.txt

```
Michael|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead
Will|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead
```

Shelley|New York|Female,27|Python:80|Test:Lead,COE:Architect

Lucy|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead

LOAD DATA LOCAL INPATH 'labs/map-reduce/hive-lab/complex-data/employee.txt'

OVERWRITE INTO TABLE employee_internal;

LOAD DATA LOCAL INPATH 'employee3.txt' OVERWRITE INTO TABLE employee_internal;

--get all the data

SELECT * FROM employee_internal;

--get work place which is array of string

SELECT name, work_place FROM employee;

SELECT name, work_place[0] AS col_1, work_place[1] AS col_2, work_place[2] AS col_3 FROM employee_internal;

-- get the gender and age structure

SELECT name, gender_age FROM employee_internal;

-- get the gender and age as separate column

SELECT name, gender_age.gender, gender_age.age FROM employee_internal;

--get the skill score

SELECT skills_score FROM employee_internal;

SELECT name, skills_score['DB'] AS DB, skills_score['Perl'] AS Perl, skills_score['Python'] AS Python, skills_score['Sales'] as Sales, skills_score['HR'] as HR FROM employee_internal;

SELECT name, depart_title['Product'] AS Product, depart_title['Test'] AS Test, depart_title['COE'] AS COE, depart_title['Sales'] AS Sales FROM employee_internal;

-- External Table

```
drop table if exists employee_external;
CREATE TABLE employee_external
(
name string,
work_place ARRAY<string>,
gender_age STRUCT<gender:string,age:smallint>,
skills_score MAP<string,smallint>,
depart_title MAP<string,ARRAY<string>>
)
COMMENT 'This is an internal table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE
LOCATION '/user/cloudera/labs/map-reduce/hive-lab/complex-data/employee.txt';

-- Other supported format
-- SEQUENCEFILE, RCFILE, ORC, AVRO
```

-- Partitioned

```
drop table if exists employee_partitioned;
CREATE TABLE employee_partitioned  (
name string,
work_place ARRAY<string>,
gender_age STRUCT<gender:string,age:smallint>,
```

```
skills_score MAP<string,int>,
depart_title MAP<STRING,ARRAY<STRING>>
 )
PARTITIONED BY (Year INT, Month INT)
ROW FORMAT DELIMITED
 FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':';

SHOW PARTITIONS employee_partitioned;
-- this does not result any data
```

The partition is not enabled automatically. We have to use ALTER TABLE ADD PARTITION to add partitions to a table.

```
ALTER TABLE employee_partitioned ADD
PARTITION (year=2017, month=10)
PARTITION (year=2017, month=11);

SHOW PARTITIONS employee_partitioned;
```

-- Drop a partition

```
ALTER TABLE employee_partitioned
DROP IF EXISTS PARTITION (year=2017, month=10);
```

Sample Data File : employee_partitioned.txt

```
Michael|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead
Will|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead
Shelley|New York|Female,27|Python:80|Test:Lead,COE:Architect
Lucy|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead
```

LOAD DATA LOCAL INPATH 'employee_partitioned.txt' OVERWRITE INTO TABLE employee_partitioned PARTITION (year=2017, month=10);

SELECT name, year, month FROM employee_partitioned;

describe formatted employee_external;

BUCKET

Sample Data File : employee_id.txt

100|Michael|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead

100|Will|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead

100|Shelley|New York|Female,27|Python:80|Test:Lead,COE:Architect

100|Lucy|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead

100|Michael2|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead

100|Will2|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead

200|Shelley2|New York|Female,27|Python:80|Test:Lead,COE:Architect

200|Lucy2|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead

200|Michael3|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead

200|Will3|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead

200|Shelley3|New York|Female,27|Python:80|Test:Lead,COE:Architect

200|Lucy3|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead

drop table if exists employee_id;

CREATE TABLE employee_id

(

emp_id int,

name string,

work_place ARRAY<string>,

```
gender_age STRUCT<gender:string,age:smallint>,
skills_score MAP<string,smallint>,
depart_title MAP<string,ARRAY<string>>
)
COMMENT 'This is an internal table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH  'employee_id.txt'  OVERWRITE INTO TABLE employee_id ;

drop table if exists employee_id_bucket;
CREATE TABLE employee_id_bucket
(
emp_id int,
name string,
work_place ARRAY<string>,
gender_age STRUCT<gender:string,age:smallint>,
skills_score MAP<string,smallint>,
depart_title MAP<string,ARRAY<string>>
)
COMMENT 'This is an internal table'
CLUSTERED BY (emp_id) INTO 2 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;
```

-- Note: if clustered column string does not match with any of the column, it may throw error;

set map.reduce.tasks = 2;

set hive.enforce.bucketing = true;

INSERT OVERWRITE TABLE employee_id_bucket SELECT * FROM employee_id;

View

CREATE VIEW employee_skills

AS

SELECT name, skills_score['DB'] AS DB,

skills_score['Perl'] AS Perl,

skills_score['Python'] AS Python,

skills_score['Sales'] as Sales,

FROM employee;

No rows affected (0.253 seconds)

Json Example

Schema of the data for JSON is

1) blogID

2) date

3) commenter name

4) comment

5) commenter email

6)commenter web site

Sample data comment.txt

{ "blogID" : "FJY26J1333", "date" : "2012-04-01", "name" : "vpxnksu", "comment" : "good stuff", "contact" : { "email" : "vpxnksu@gmail.com", "website" : "vpxnksu.wordpress.com" } }

{ "blogID" : "FJY26J1333", "date" : "2012-04-01", "name" : "vpxnksu", "comment" : "good stuff", "contact" : { "email" : "vpxnksu@gmail.com", "website" : "vpxnksu.wordpress.com" } }

{ "blogID" : "VSAUMDFGSD", "date" : "2012-04-01", "name" : "yhftrcx", "comment" : "another comment",}

```
CREATE  EXTERNAL  TABLE comments_external

(

cmt STRING

)

COMMENT 'This is an external table'

ROW FORMAT DELIMITED

STORED AS TEXTFILE

LOCATION  '/user/test/comment';


LOAD DATA LOCAL INPATH 'comment.txt' OVERWRITE INTO TABLE comments_external;
```

```
SELECT b.blogID, c.email FROM comments_external a LATERAL VIEW json_tuple('blogID', 'contact') b
AS blogID, contact  LATERAL VIEW json_tuple(b.contact, 'email', 'website') c
AS email, website WHERE b.blogID='64FY4D0B28';
```

PySpark

Spark Submit Example

```
#import necessary functions at the begining of the program
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

#standard varialbes
APP_NAME = "My-Simple-PySpark-Program"


def main(spark_session):
    # Have you processing here
    # Create a datafram using range function which start with 1 and ends with 100 with 2 as step (1,3,
5 ...)
    df =spark_session.range(1,100,2)
    df.printSchema()
    df.show()

if __name__ == '__main__':
    print("The program started... ")
    #Create The spark session by giving app name

    #https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.SparkSession
    spark = SparkSession.builder.appName(APP_NAME).getOrCreate()

    main(spark)
    print("Program is over")
```

Reading Json File

spark.read.json("file:///tmp/tips/data/flight-data.json")

'''
DataFrame[DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string, count: bigint]
'''

tips_df = spark.read.json("file:///tmp/tips/data/flight-data.json")
tips_df.printSchema()

'''
root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: long (nullable = true)
'''

tips_df.show()

'''
+-------------------+-------------------+-----+
| DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-------------------+-------------------+-----+
United States	Romania	15
United States	Croatia	1
United States	Ireland	344
Egypt	United States	15
United States	India	62
United States	Singapore	1
United States	Grenada	62
Costa Rica	United States	588
Senegal	United States	40
Moldova	United States	1
United States	Sint Maarten	325
United States	Marshall Islands	39
Guyana	United States	64
Malta	United States	1
Anguilla	United States	41
Bolivia	United States	30
United States	Paraguay	6
Algeria	United States	4
Turks and Caicos ...	United States	230

```
|      United States|         Gibraltar|    1|
+-------------------+------------------+-----+
only showing top 20 rows
'''
```

tips_df =
spark.read.format("json").option("mode","FAILFAST").option("inferSchema","true").load("file:///tmp/tips/data/flight-data.json")
tips_df.printSchema()

```
'''
root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: long (nullable = true)
'''
```

tips_df.createOrReplaceTempVeiw("tips_table")
tips_df.createOrReplaceTempView("tips_table")
spark.sql("select count from tips_table").show(5)

```
'''
20/05/07 10:19:55 WARN ObjectStore: Version information not found in metastore.
hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
20/05/07 10:19:55 WARN ObjectStore: Failed to get database default, returning
NoSuchObjectException
20/05/07 10:19:56 WARN ObjectStore: Failed to get database global_temp, returning
NoSuchObjectException
+-----+
|count|
+-----+
|   15|
|    1|
|  344|
|   15|
|   62|
+-----+
only showing top 5 rows
'''
```

Rearrange Column In Dataframe

```
#import necessary files
from pyspark.sql.types import StructType, StructField, IntegerType, StringType


tips_list = [[1,"item 1" , 2], [2 ,"item 2", 3],  [1 ,"item 3" ,2],  [3 ,"item 4" , 5]]
tips_schema = StructType(  [StructField("Item_ID",IntegerType(),
True),StructField("Item_Name",StringType(), True ),  StructField("Quantity",IntegerType(), True)])

#create dataframe using spark session
tips_df = spark.createDataFrame(tips_list, tips_schema)

#the schema
tips_df.printSchema()


'''
root
 |-- Item_ID: integer (nullable = true)
 |-- Item_Name: string (nullable = true)
 |-- Quantity: integer (nullable = true)
'''

#register as table
tips_df.createOrReplaceTempView("tips_table")

tips_new_df = spark.sql("select Item_Name, Item_ID, Quantity from tips_table")

tips_new_df.printSchema()

'''
root
 |-- Item_Name: string (nullable = true)
 |-- Item_ID: integer (nullable = true)
 |-- Quantity: integer (nullable = true)
'''
#using the dataframe approach

#create dataframe using spark session
tips_df = spark.createDataFrame(tips_list, tips_schema)

#use the select function from the dataframe
tips_new_df = tips_df.select("Item_Name","Item_ID","Quantity")
tips_new_df.printSchema()
```