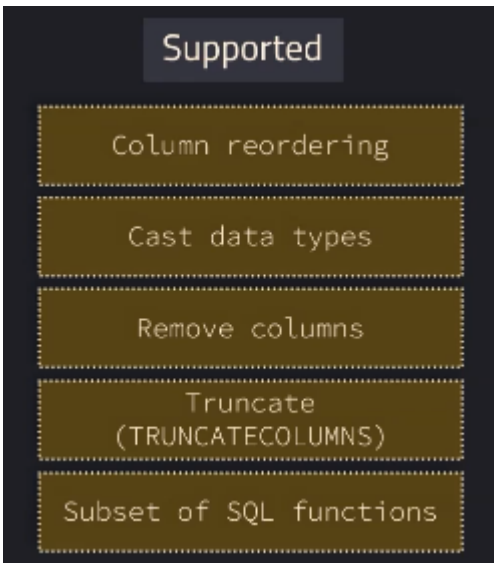
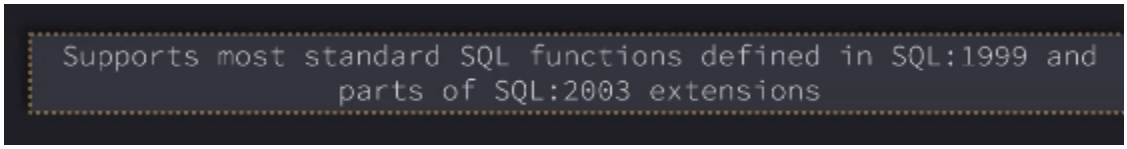
Transformations and Functions

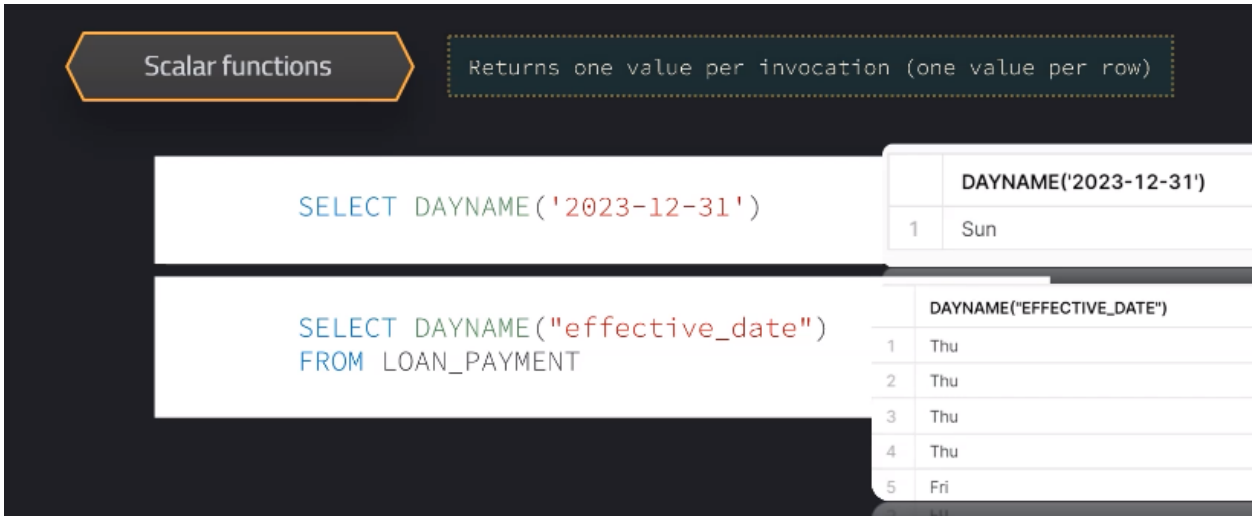
1. When loading the data ..we can apply some transformations
2. So when we want to load data, we can also, instead of just using an entire file, we can query and specify a specific select statement in the copy command.
3. What we can do is 

4. What we cannot do 

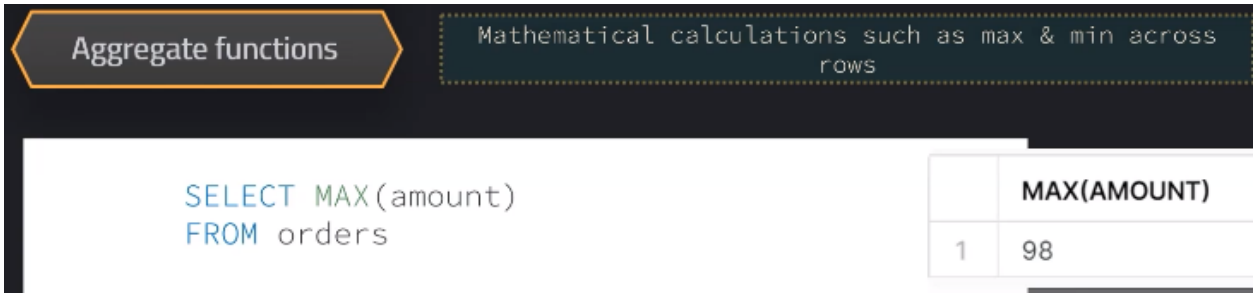
5. But in general we can say that there are a lot of functions available when we want to query data in Snowflake so we can apply and use functions in Snowflake. And most of the SQL functions that are defined in SQL 1999 are also supported.



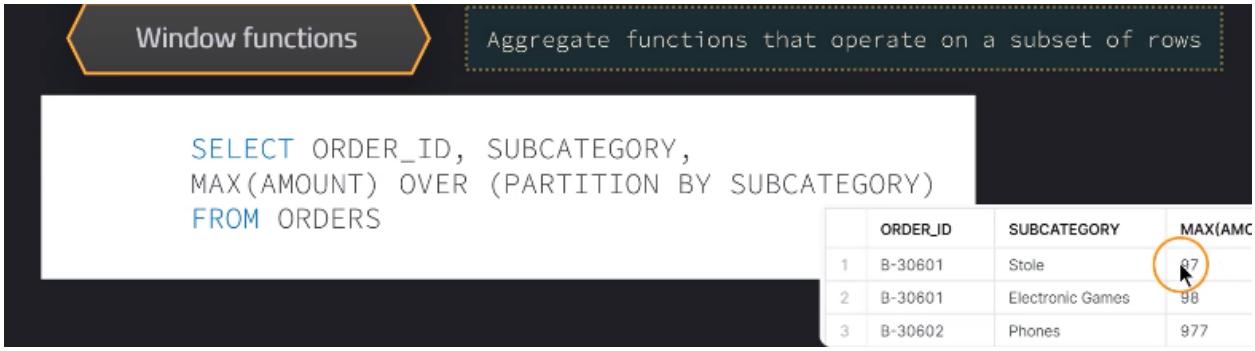
6. Scalar functions

7. 

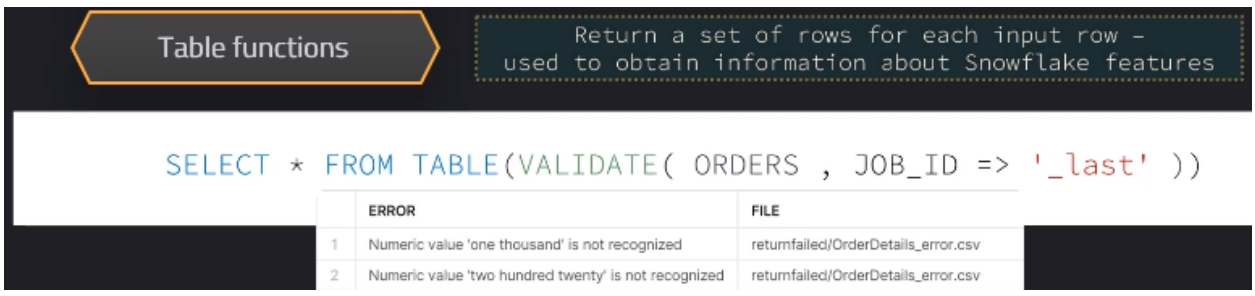
8. Aggregate functions



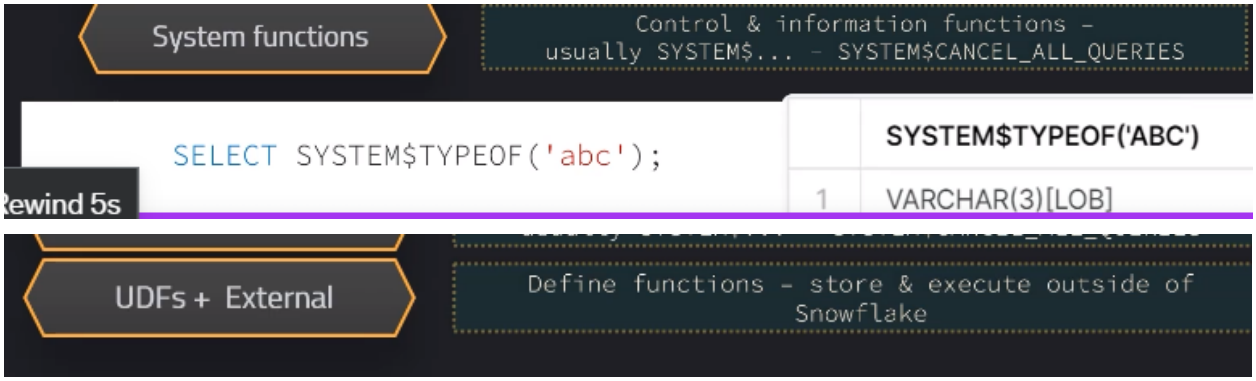
9. Window functions



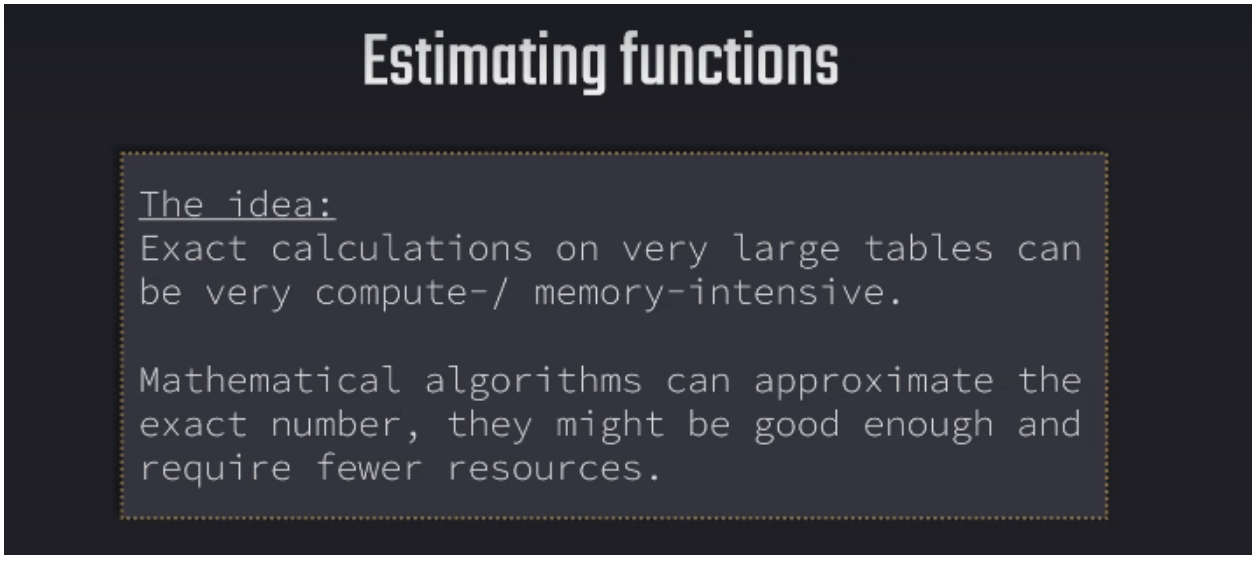
10. Table Functions

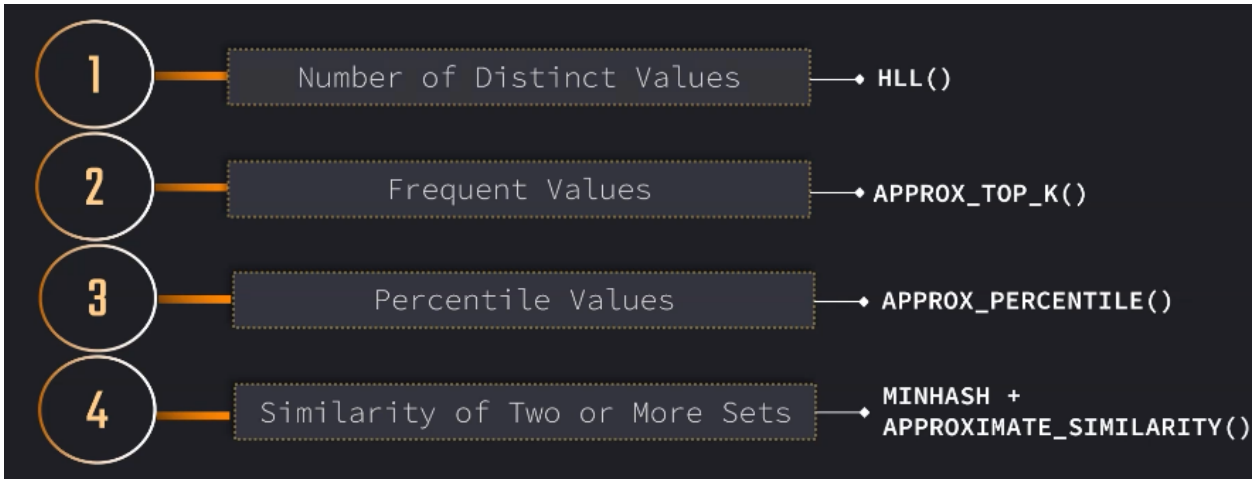


11. Table functions : https://docs.snowflake.com/en/sql-reference/functions-table

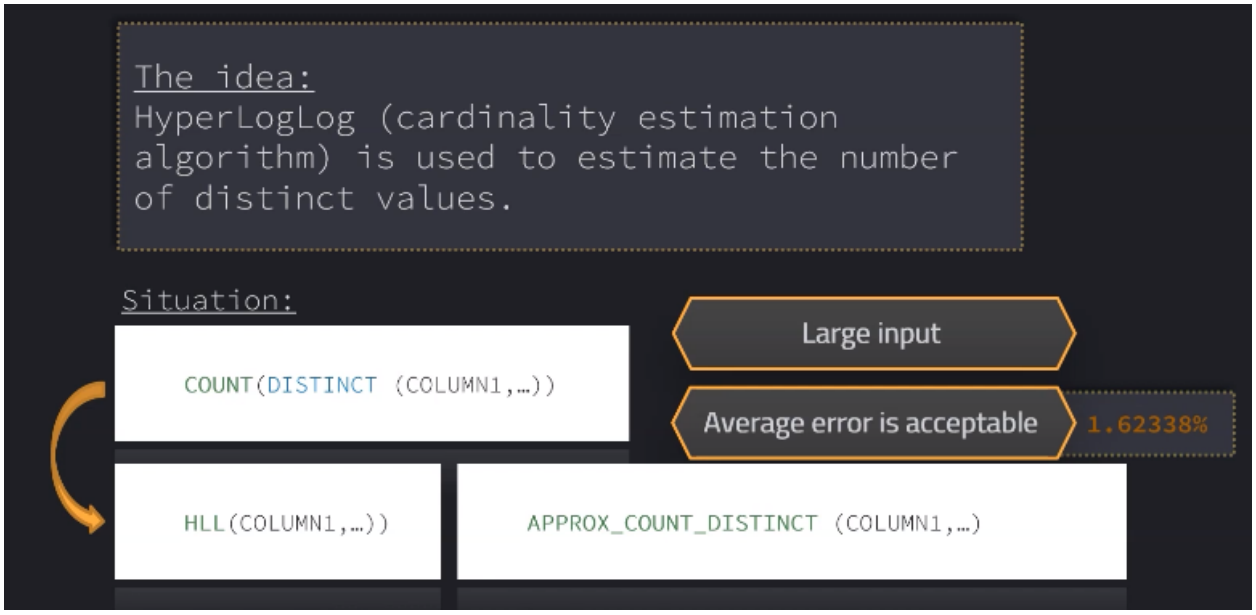
12. System function
13. 

Estimation Functions

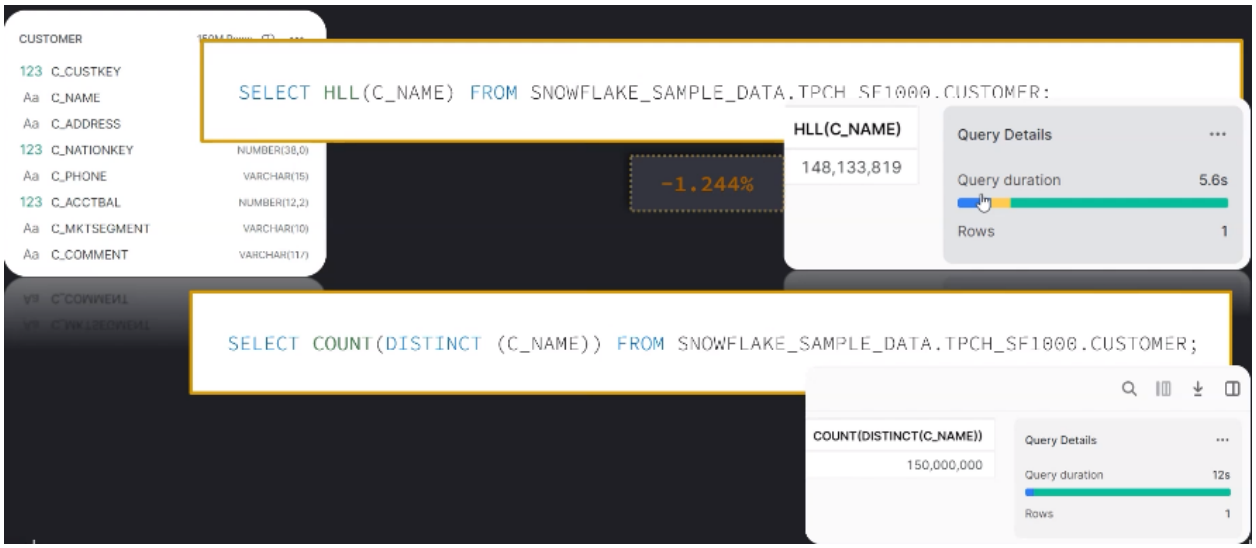
1. 

2. Here we have 4 types of functions


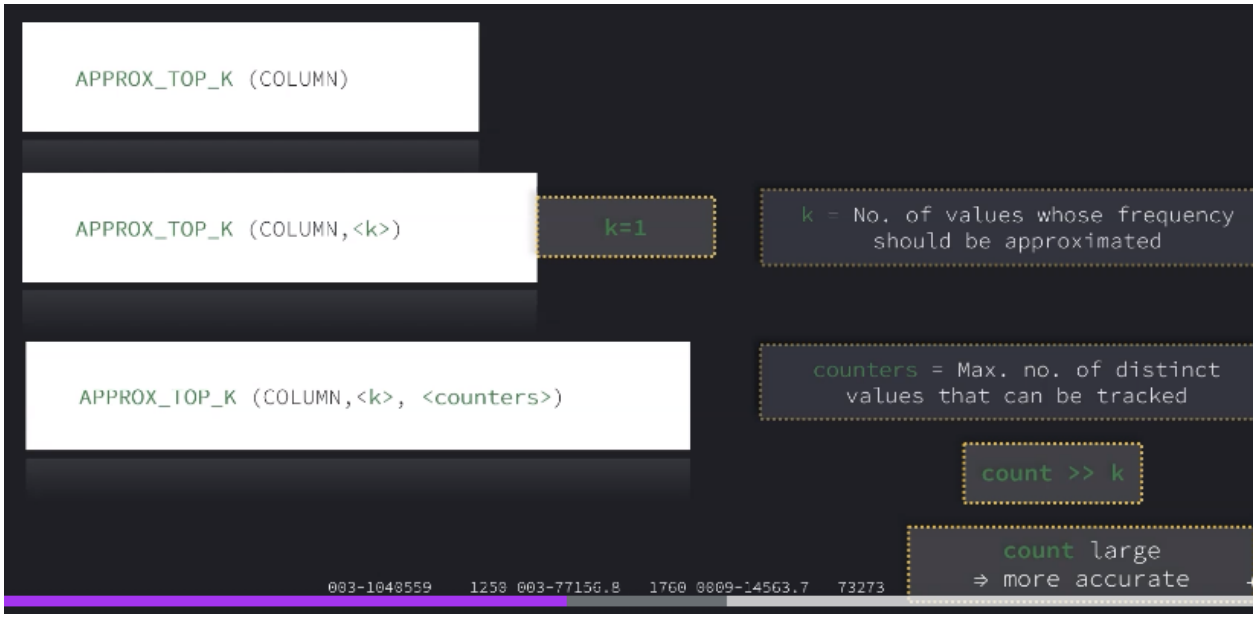
3. HLL

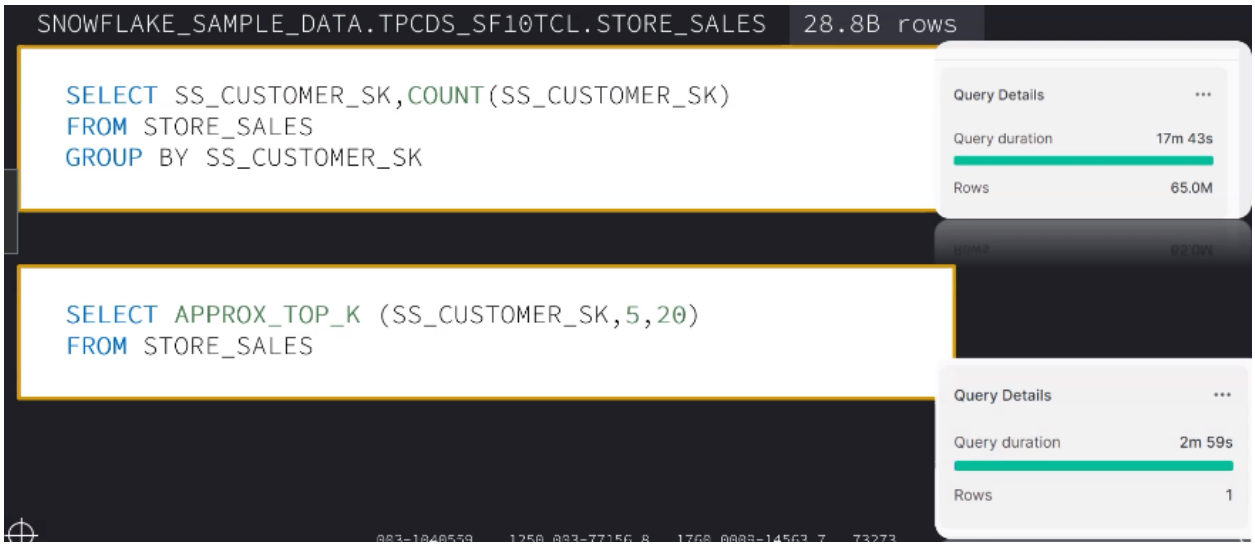
4. 

5. HLL gives approximate count of distinct columns in our table
6. Difference between using HLL and count …with a table of 1.5mil rows



7. Frequent values

8. 

9. Here if k = 10..returns top 10 frequent values
10. Example ..here using
11. 

12. Result form aprrox_top_k..will look like an list of list

	APPROX_TOP_K (SS_CUSTOMER_SK,5,20)
1	[[20952969, 174160540], [44412221, 174160540], [30221701, 174160530], [30594069, 174160530], [53273150, 174160530]]

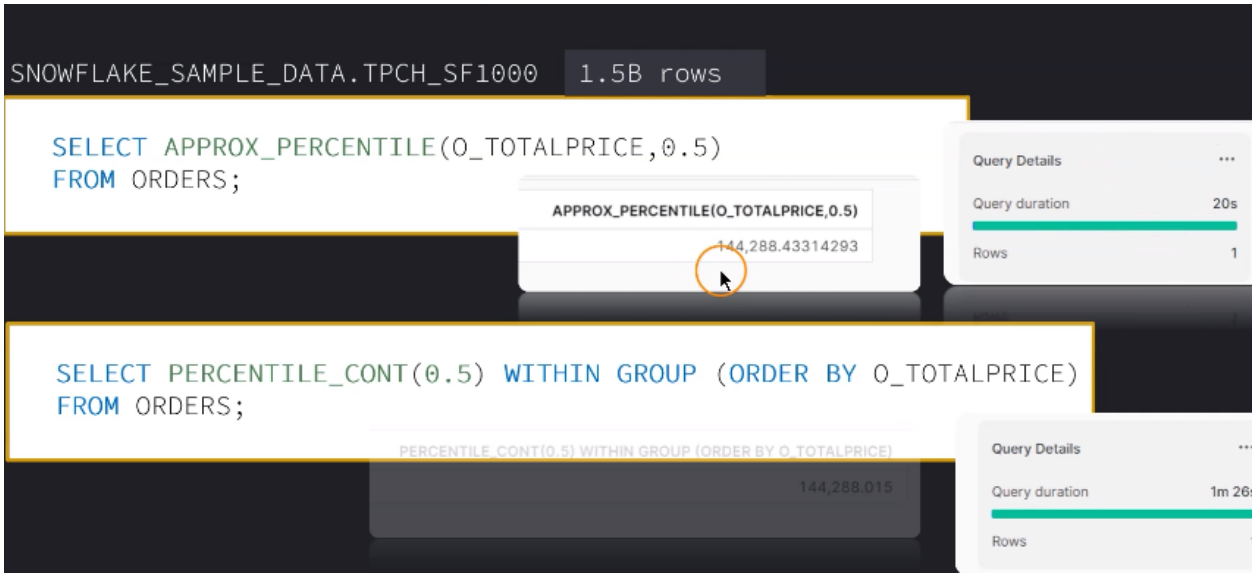
13. Percentile value

14. Percentile_cont in sql

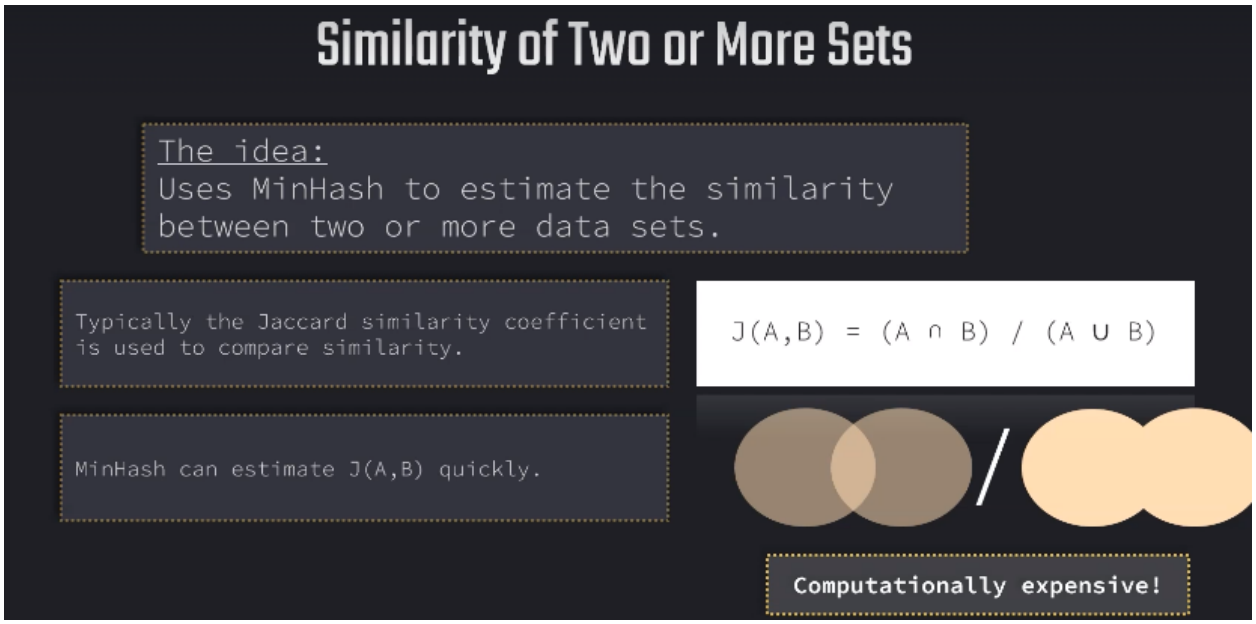
The `PERCENTILE_CONT()` function in SQL is an aggregate function that returns the value at a specified percentile within a group of rows. The percentile value is a number between 0 and 1, where 0 represents the lowest value in the group and 1 represents the highest value.

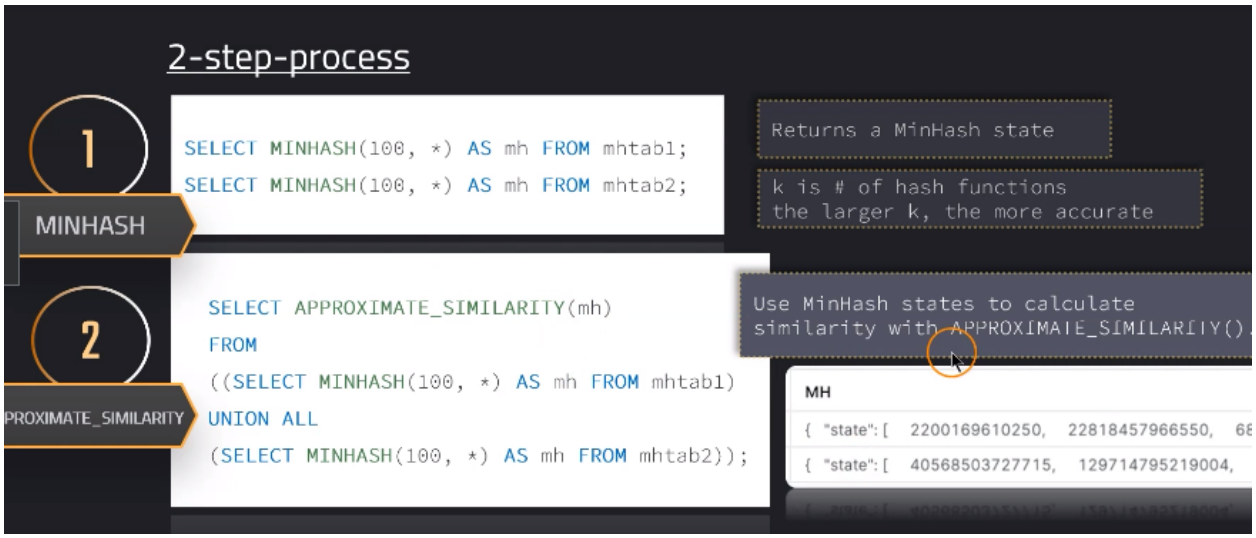
The `PERCENTILE_CONT()` function uses linear interpolation to calculate the percentile value. This means that if the percentile value falls between two rows in the group, the function will return a weighted average of the values of those two rows.

15. Diff bw percentile cont and approx_percentile



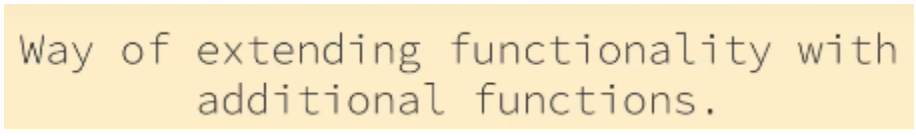
16. Similarity between two data sets

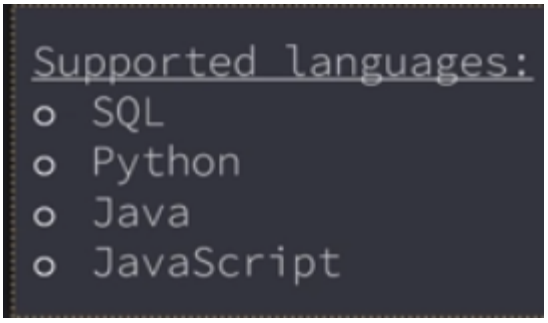
17. 

18. Snowflake implemented MinHash..which lowers computation required
19. 

20. Refer online too
21. Result of that 2nd query would be 0 or 1..0 means no similarity…1 means similarity

User-defined Functions

1. 

2. It supports writing in 

3. Writing an UDFs using SQL (add_two function)
4. First we create Function with the parameter..and we shud specify the datatype of parameter..then next we have to return datatype..what the function returns

```
create function add_two(n int)
returns int
```

5. Then we use “as” to define our func..and two $$ represents the body of our logic

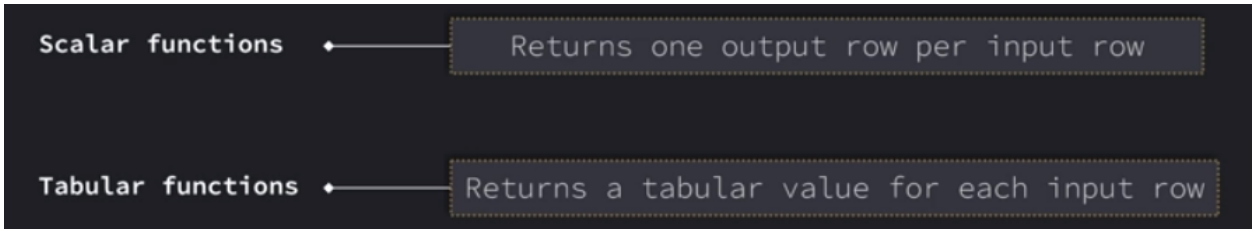
```
create function add_two(n int)
returns int
as
$$
n+2
$$;
```

	status
1	Function ADD_TWO successfully created.

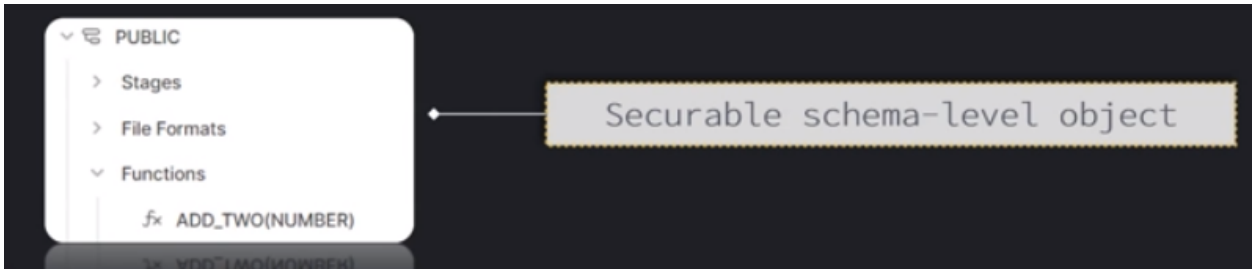
6. To use python..first two lines would be same for every language
7. Then if we are using python…we have to specify language,version and handler
8. And in the $$(body) we have to use python function

```
create function add_two(n int)
returns int
language python
runtime_version = '3.8'
handler = 'addtwo'
as
$$
def add_two(n):
return n+2
$$;
```

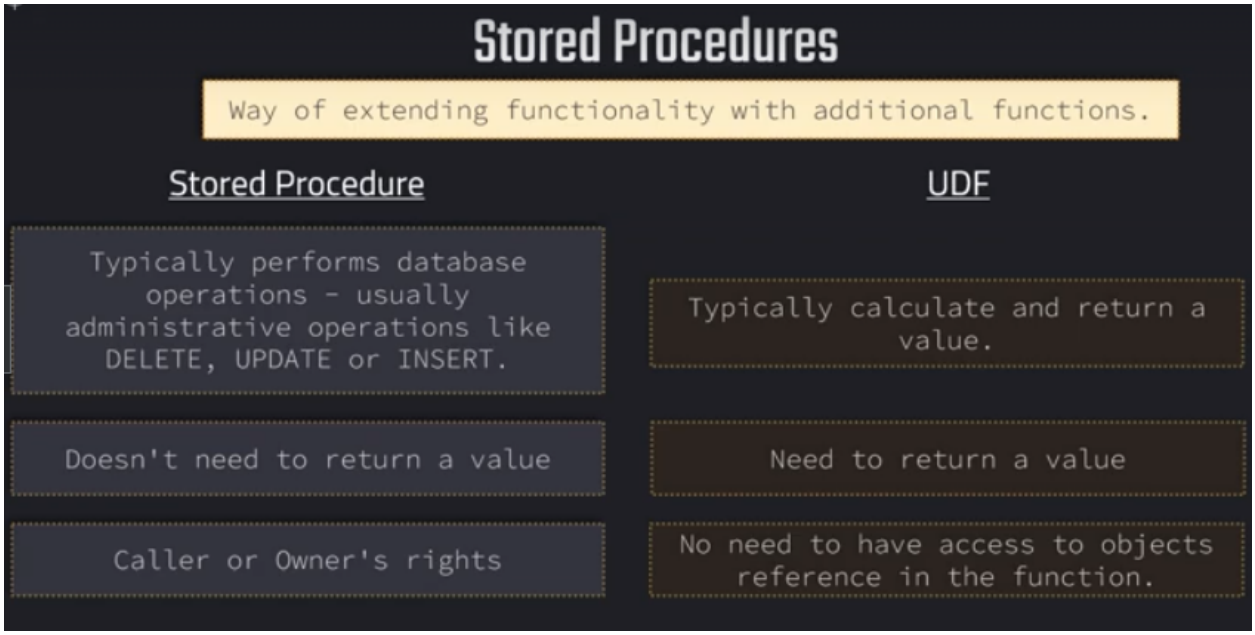
9. We have two types of function in UDF



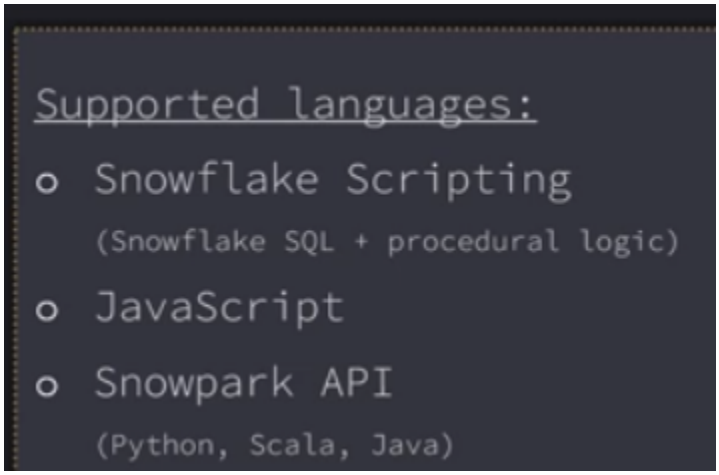
10. The functions are schema level objects

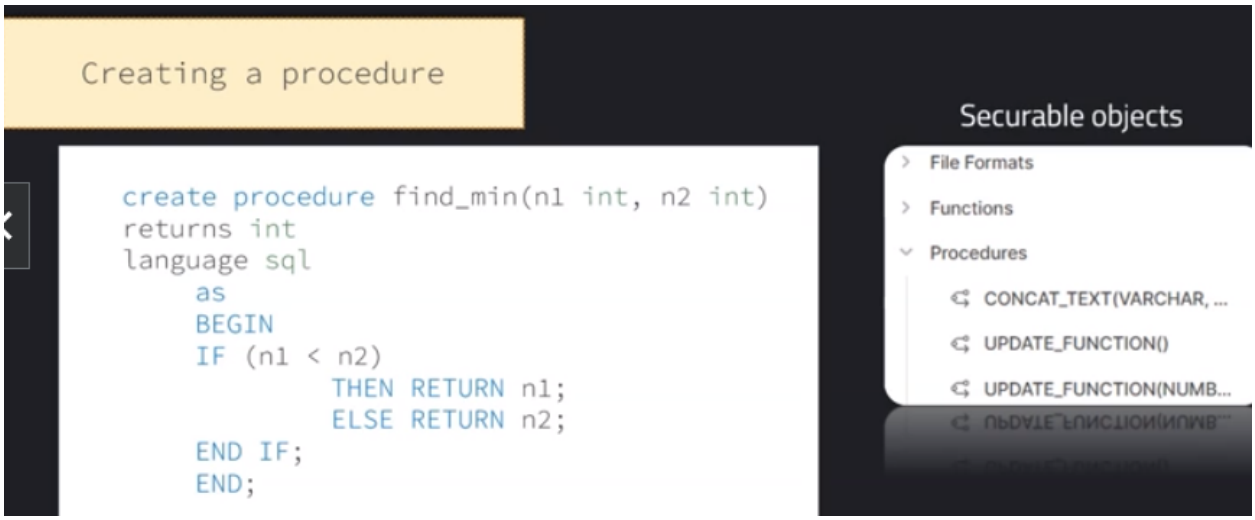


Stored Procedure

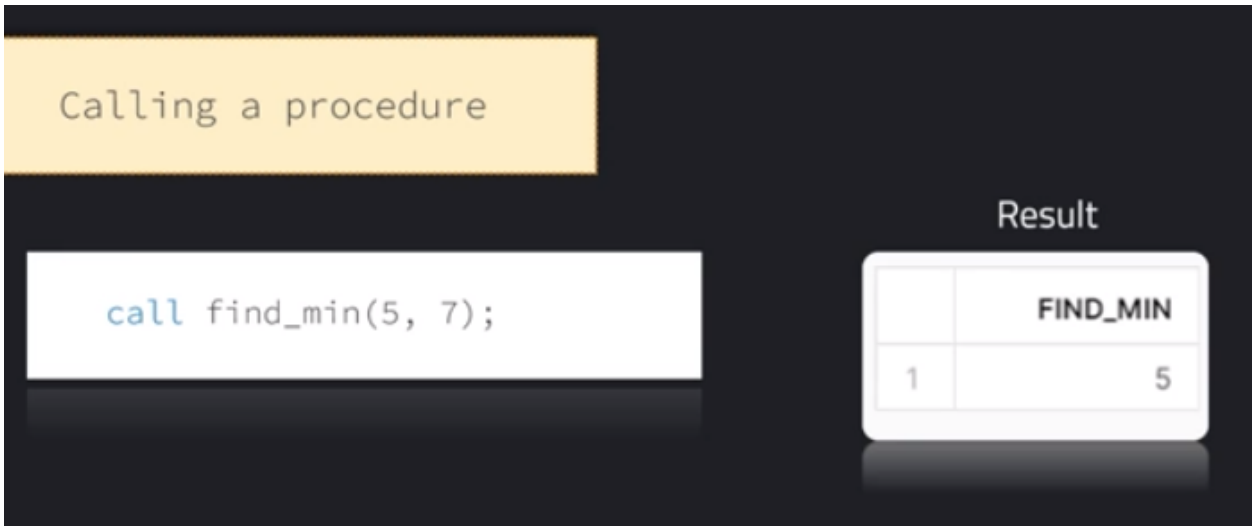
1. 

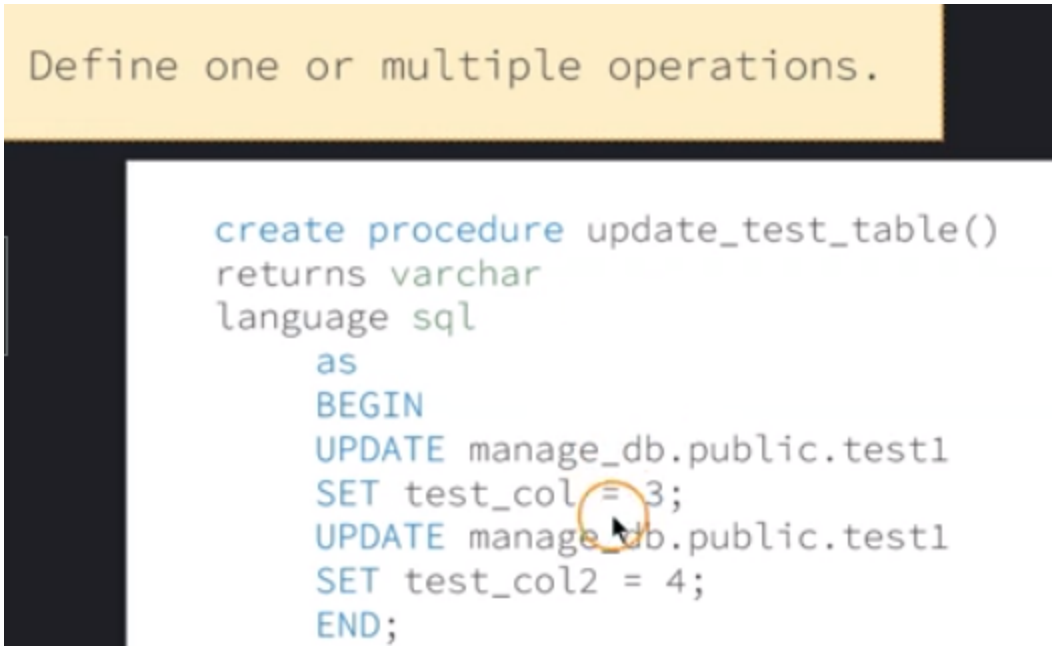
2. Supported languages for stored procedures in Snowflake

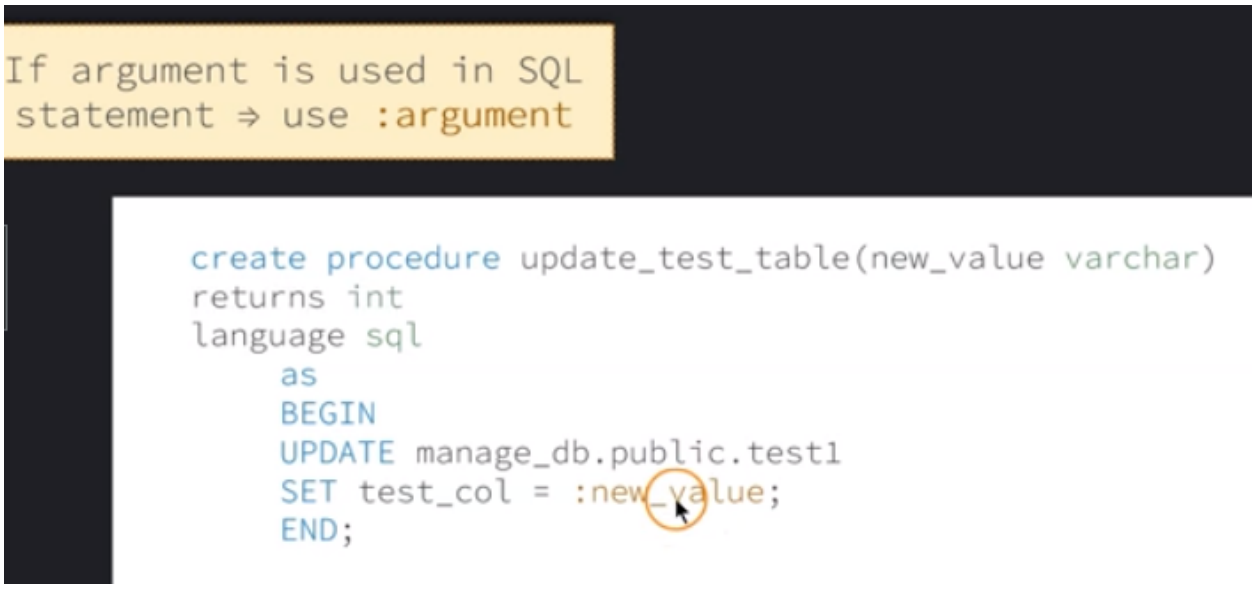


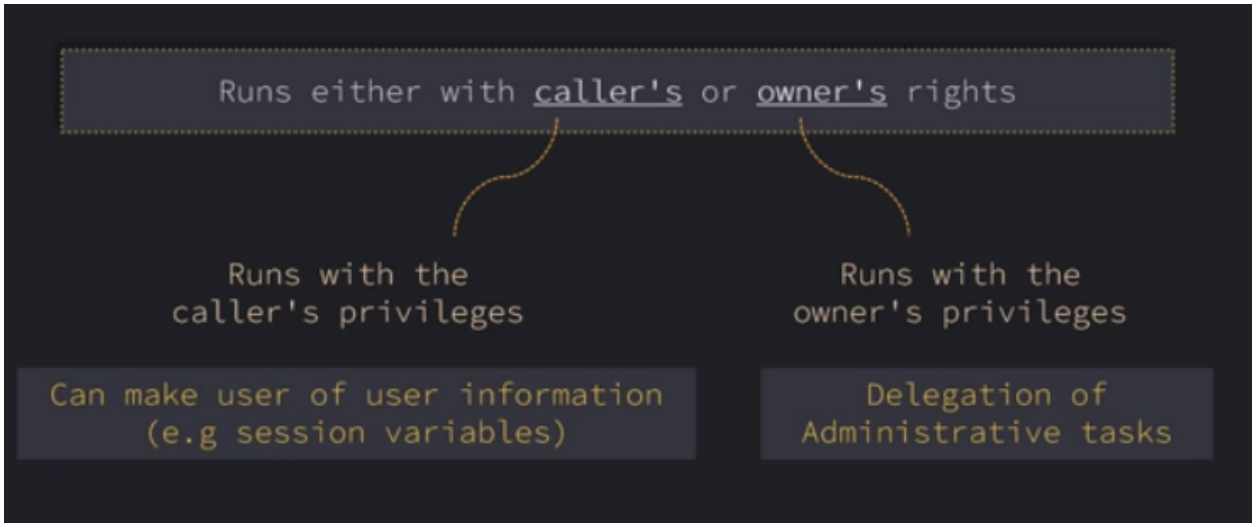
3. 

4. It is also a secure schema object..that means..we can give permission to users to use our stored procedure

5. 

6. 
Define one or multiple operations.
create procedure update_test_table()
returns varchar
language sql
as
BEGIN
UPDATE manage_db.public.test1
SET test_col = 3;
UPDATE manage_db.public.test1
SET test_col2 = 4;
END;

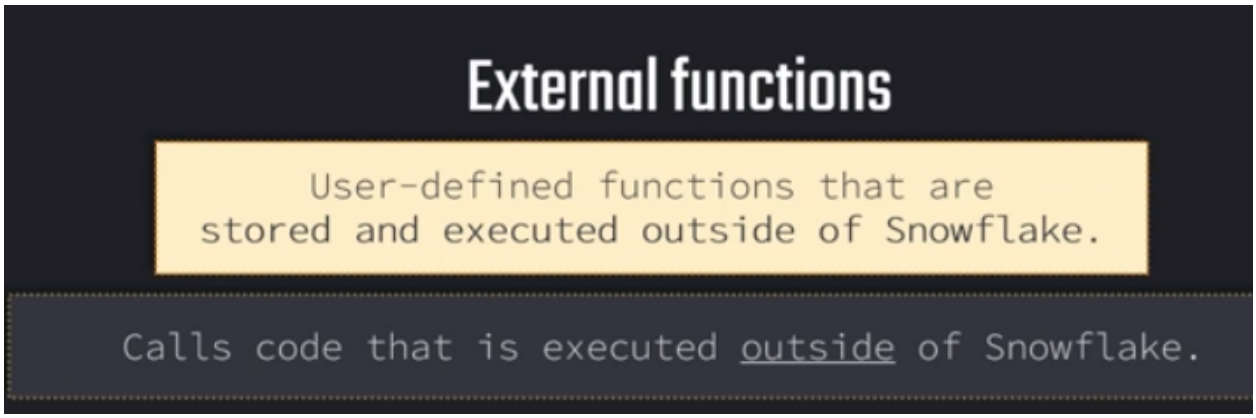
7. 



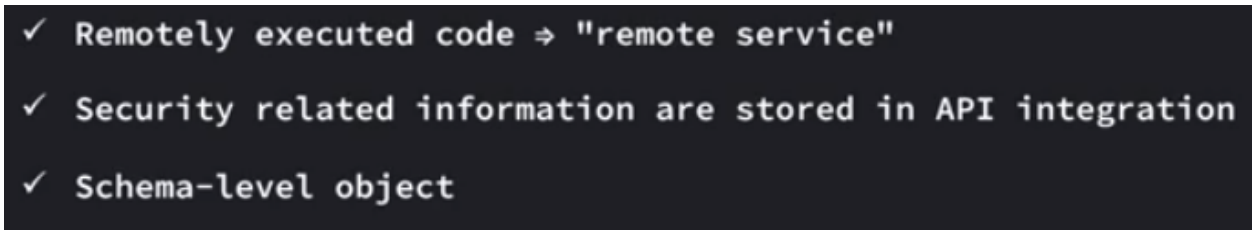
8. 

9. Should learn more about snowflake stored procedure

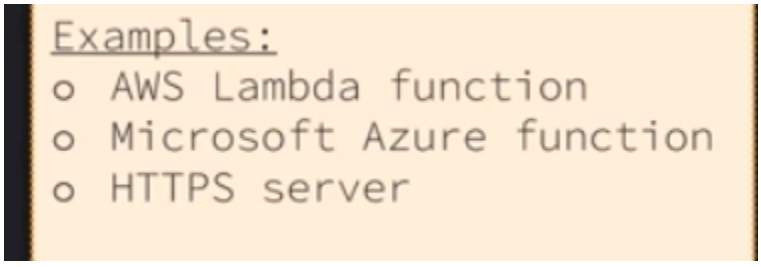
External Functions

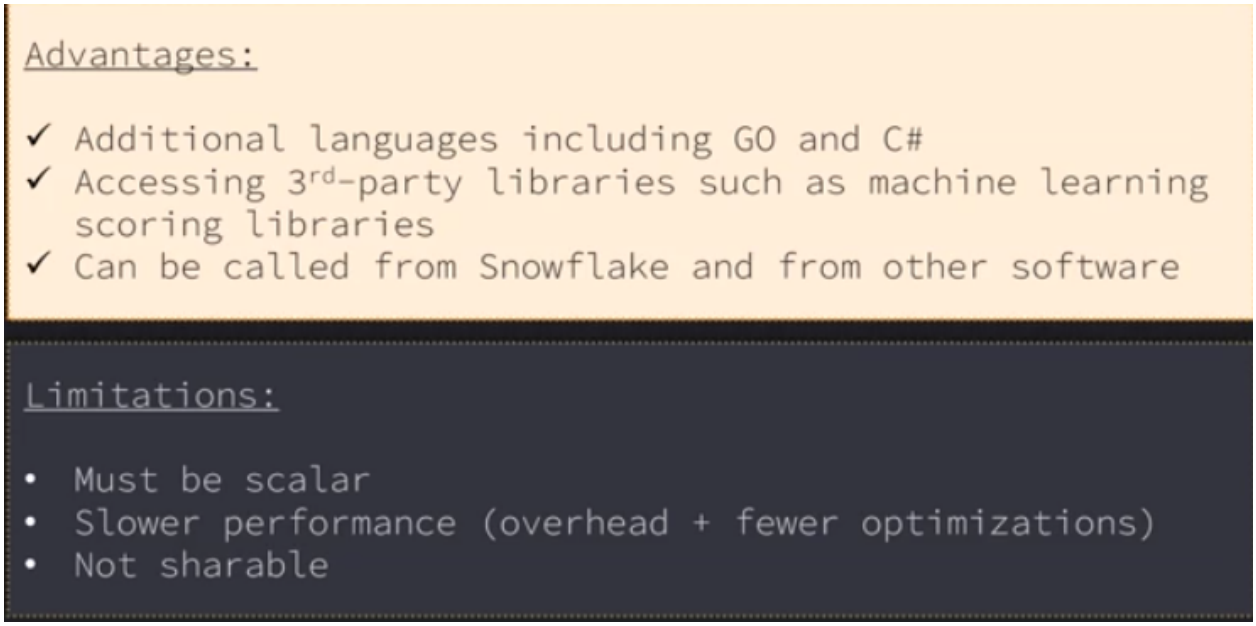
1. 

2. So for example, it will be some Azure function that we can just reference in our external function and it will be then executed outside of Snowflake.
3. 

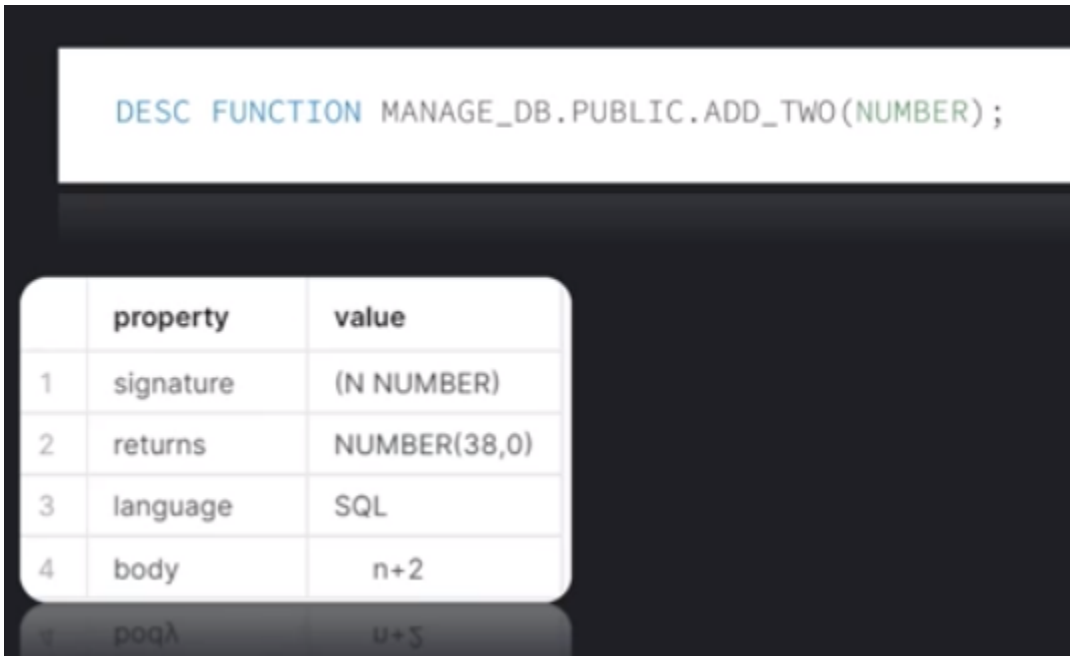
4. 

5. This can be included in external functions



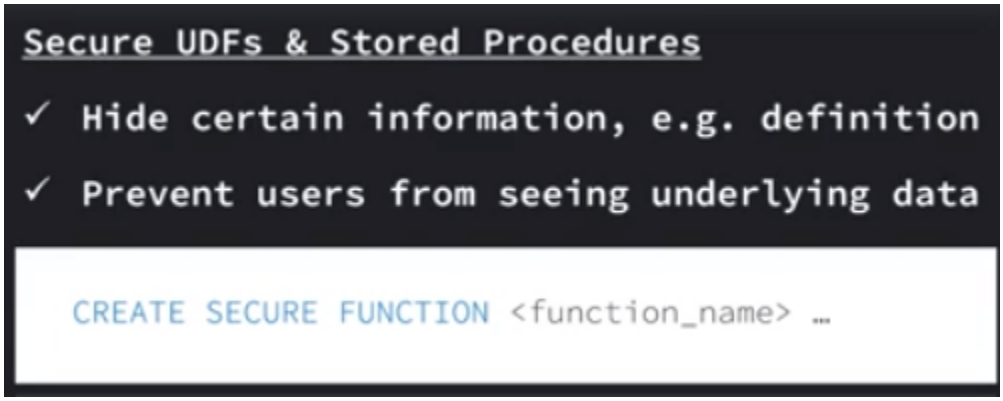
6. 

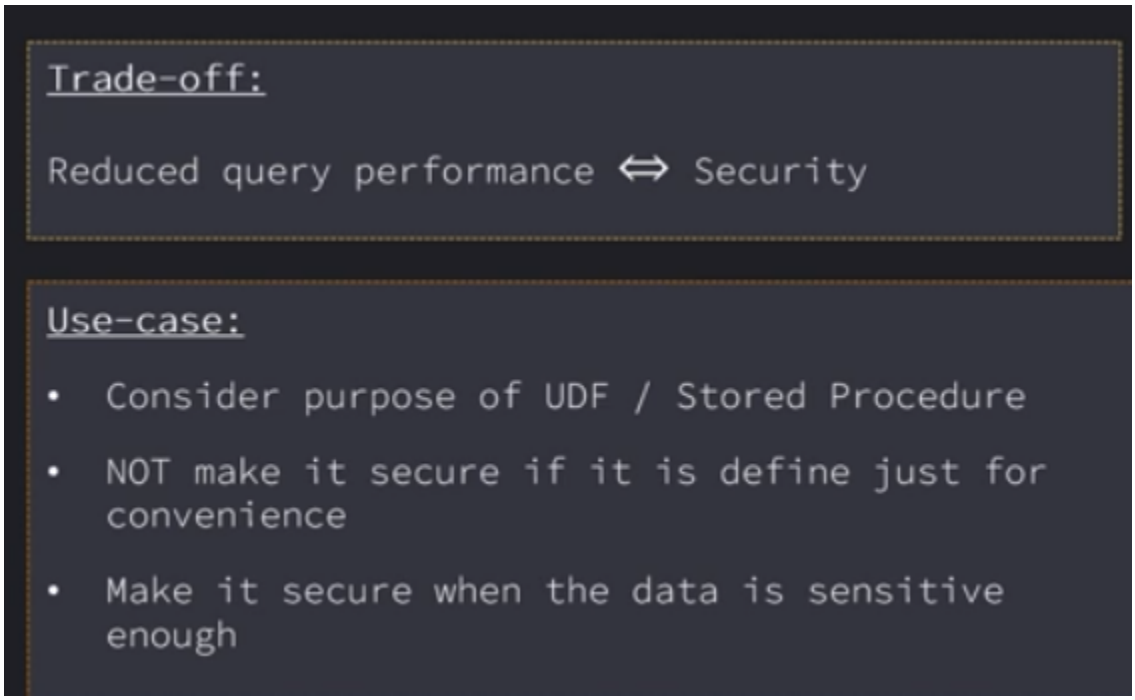
Secure UDFs & Stored Procedure

1. It may be possible that UDF or stored procedure..may contain some sensitive information
2. Some user can use desc command to see some partial information about the table and its structure
3. 

4. The other way to access this underlying data indirectly is by using optimizer

5. So to hide those information..we have to use secure UDF

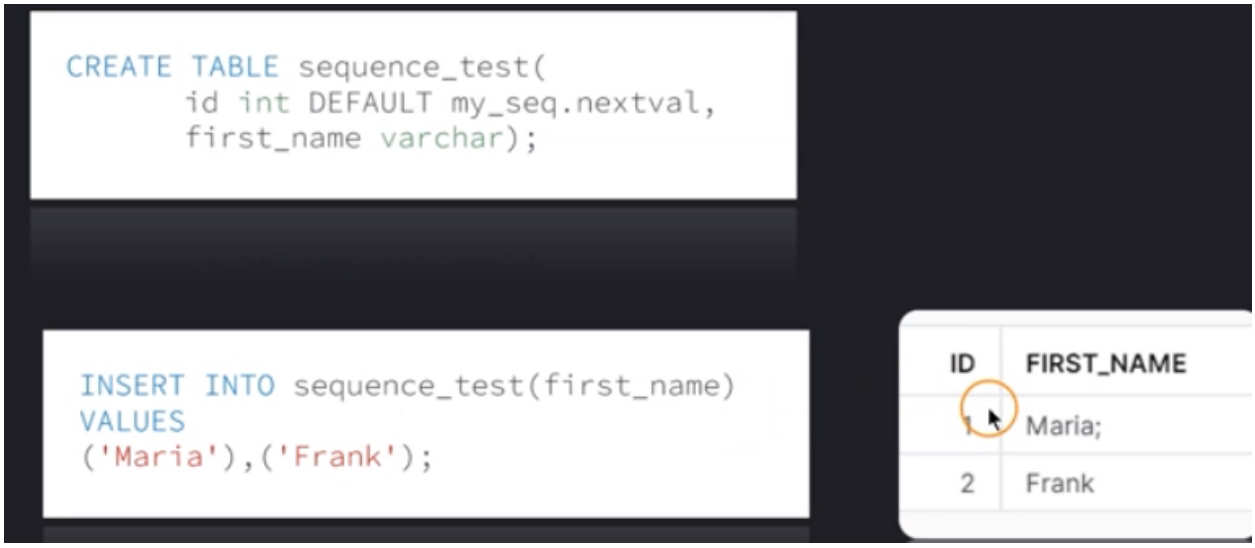


6. 

Sequences

1. A sequence database object in Snowflake is a special type of object that is used to generate unique numbers. Sequences are similar to an auto-incrementing field in a database table, but more powerful and flexible. A sequence object can be used to generate numbers in a specific range, with a specific increment, and with a specific starting point.

   Sequences are often used to generate unique primary keys for database tables. They can also be used to generate unique identifiers for other types of data, such as customer numbers, order numbers, and transaction numbers.

2. 

3. Hands on first we create sequence

```
CREATE OR REPLACE SEQUENCE my_seq
START = 1
INCREMENT = 1;
```

```
SELECT my_seq.nextval;
SELECT my_seq.nextval,my_seq.nextval;
```

4. Now if runs this 2 queries
5. For first query we get 1…for 2nd query we get 2 and 3 as output
6. Values of the sequence may not be consecutive..like 1,2,3,4..etc

7. Now we create table with id (int) and first_name ..for the id we gave sequence

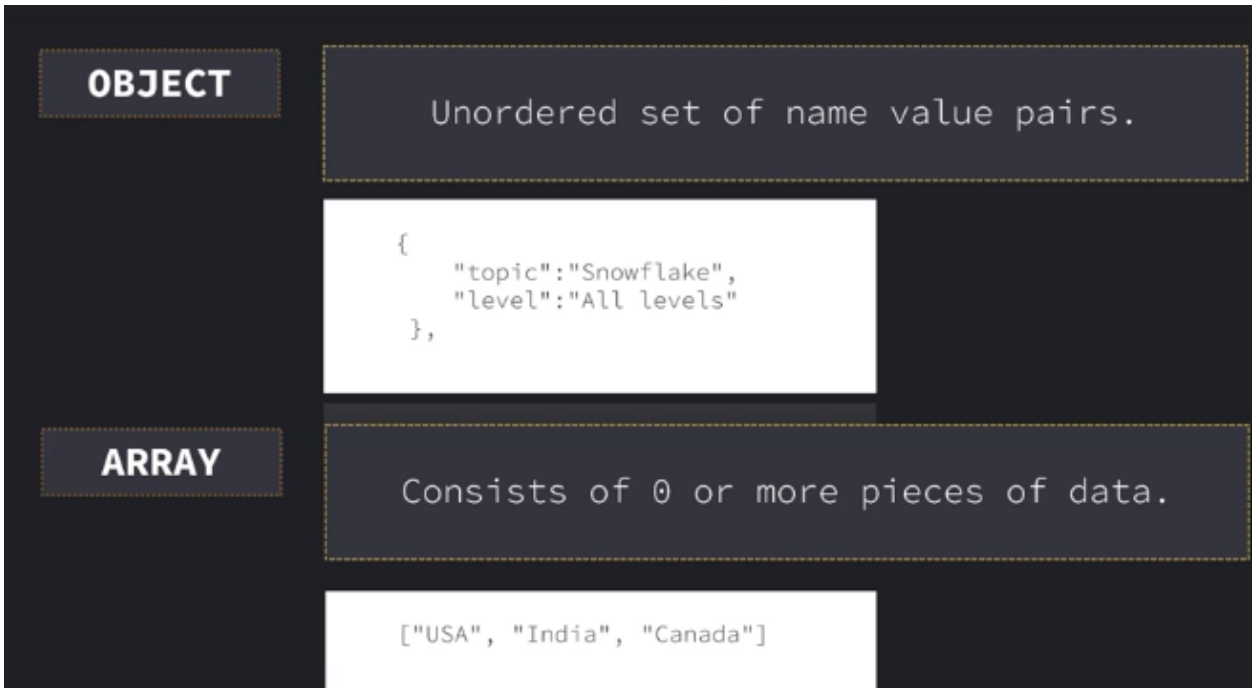
```
CREATE OR REPLACE TABLE sequence_test(
id int DEFAULT my_seq.nextval,
first_name varchar);

INSERT INTO sequence_test(first_name)
VALUES
('Maria'),
('Frank');

SELECT * FROM sequence_test;
```

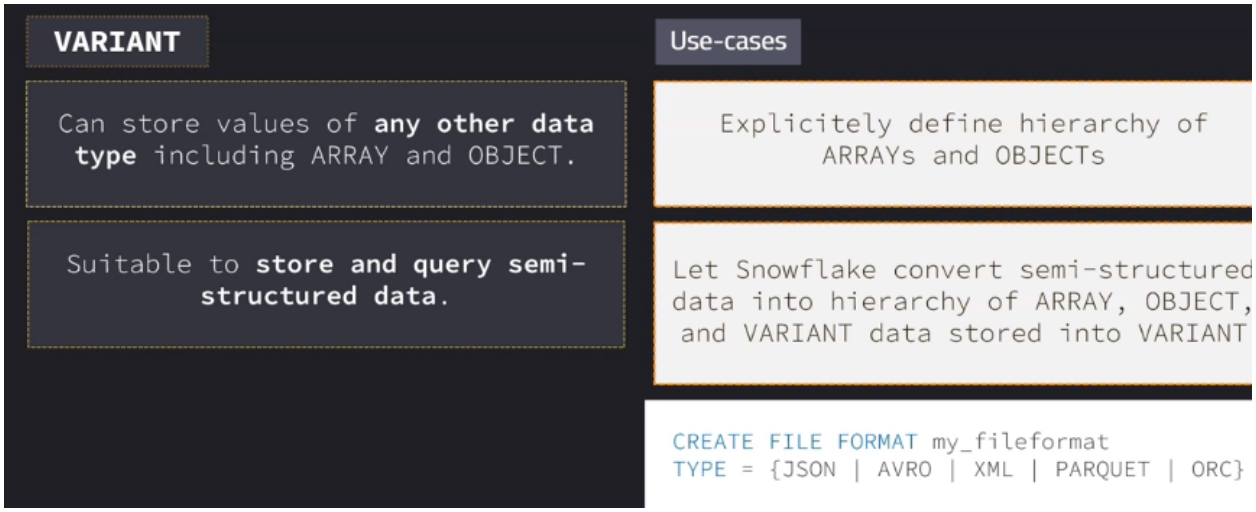
8. Now if insert values into the table..id will get random numbers from mySeq

SemiStructured Data

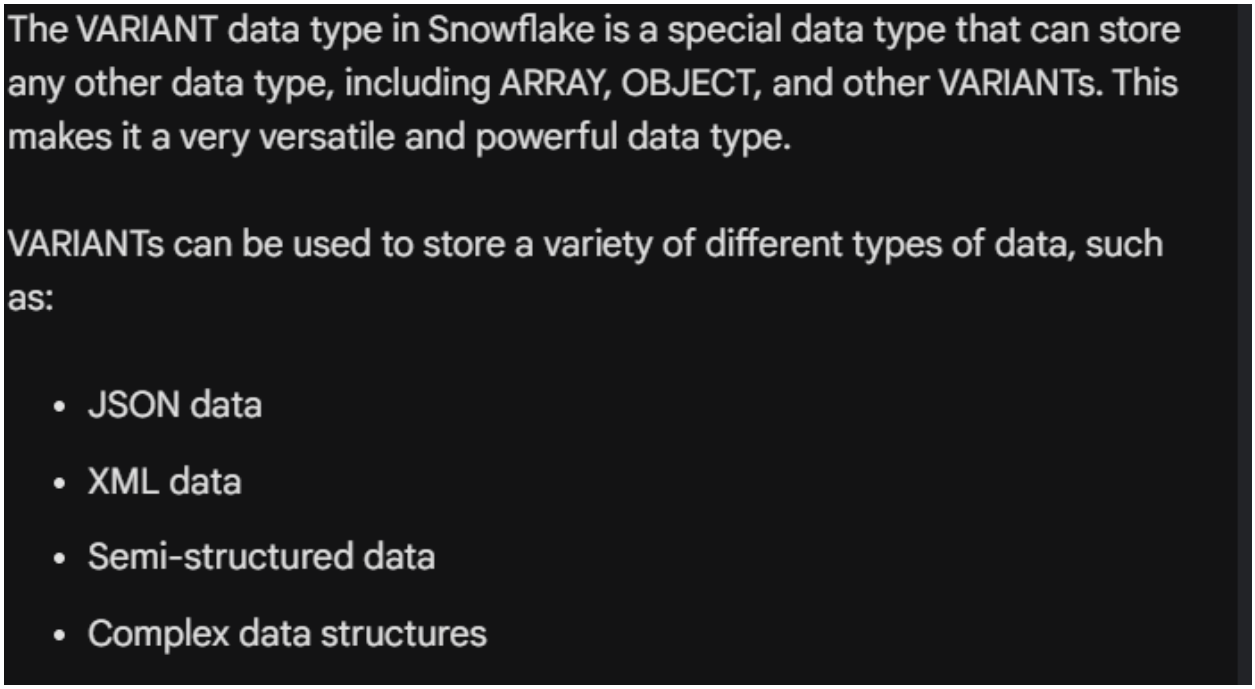
1. Very imp to know how snowflake tackles semistructured data
2. Because Snowflake actually has really great native support for these type of data formats.
3. Examples of structured and semistruc data

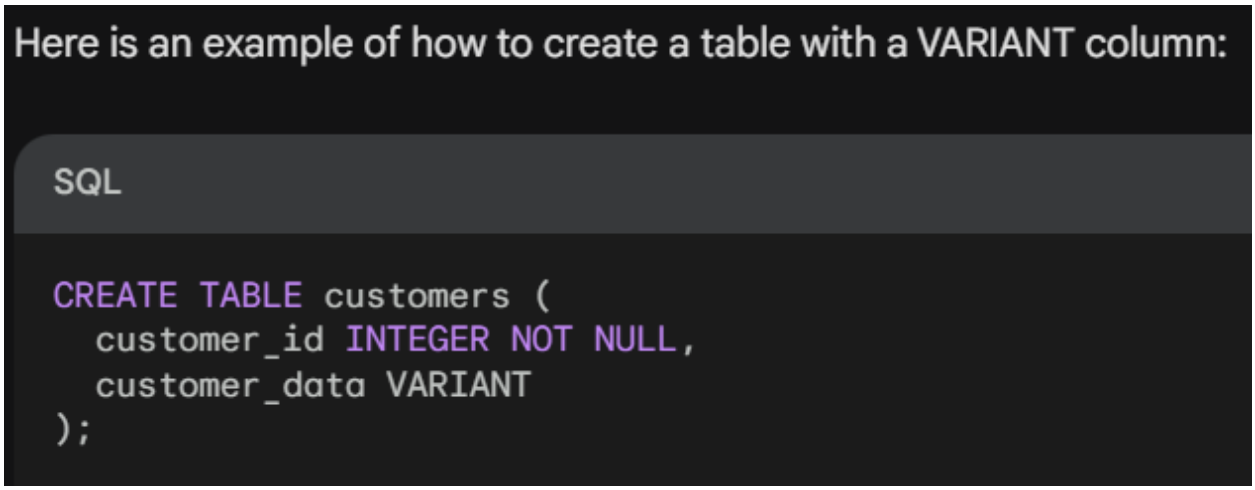
4. Supported format for semiStructures
5. We have 3 data types in snowflake semistruc



6. 
VARIANT
Can store values of any other data type including ARRAY and OBJECT.
Suitable to store and query semi-structured data.
Use-cases
Explicitely define hierarchy of ARRAYs and OBJECTs
Let Snowflake convert semi-structured data into hierarchy of ARRAY, OBJECT, and VARIANT data stored into VARIANT
CREATE FILE FORMAT my_fileformat
TYPE = {JSON | AVRO | XML | PARQUET | ORC}

7. And third is variant data type…which store all the other data types including itself





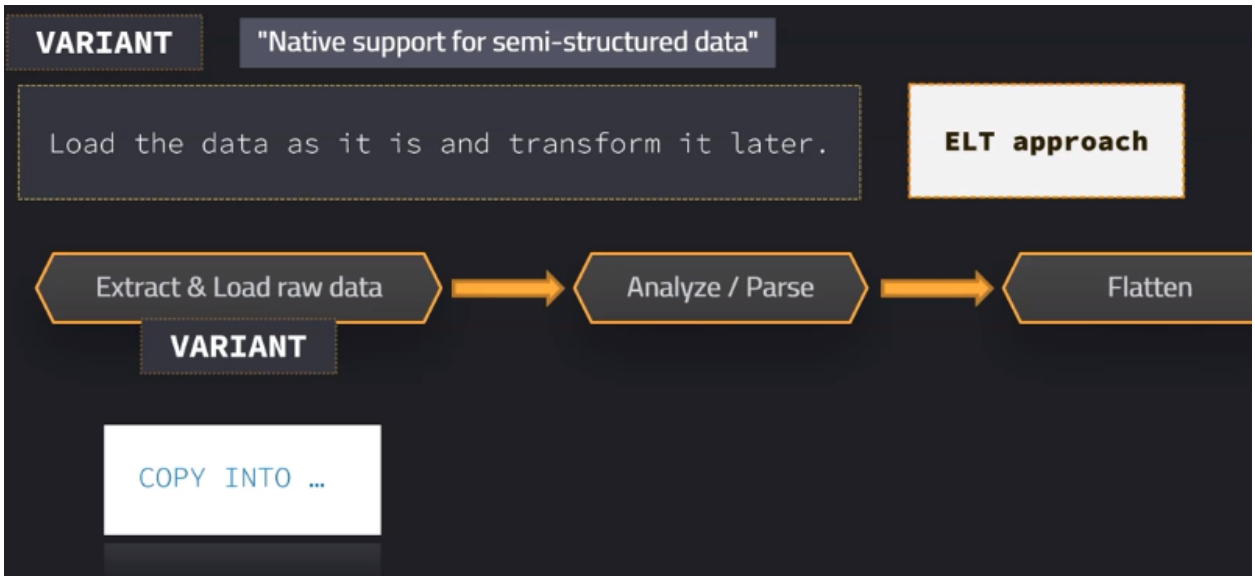
8. INSERT INTO customers (customer_id, customer_data) VALUES (12345, JSON_PARSE('{ "name": "John Doe", "email": "john.doe@example.com" }'));

9.
```sql
SELECT customer_data
FROM customers;
```

Use code with caution. Learn more

This will return the JSON object that was inserted into the `customer_data` column.

10. Variant will load the data as it is..and we will transform this data later(ELT)



Hands on- semi structured data

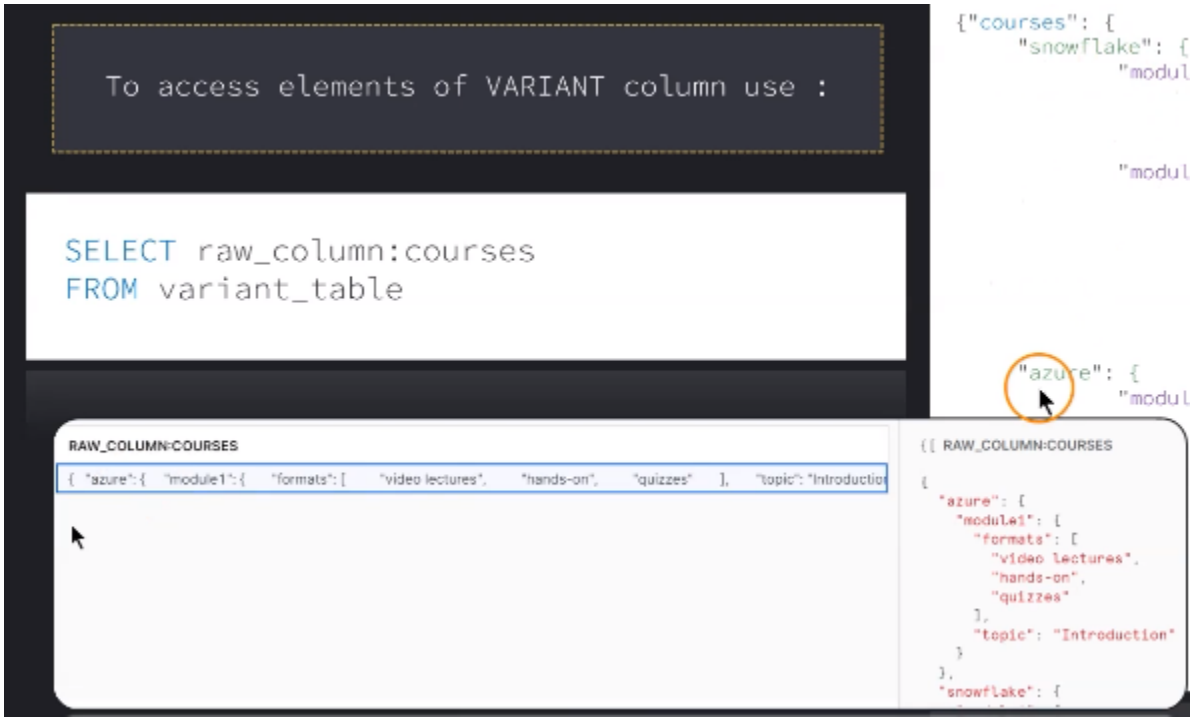
1. Refer online

Query semi structured data

1. Lets assume we have this kind of data in variant column

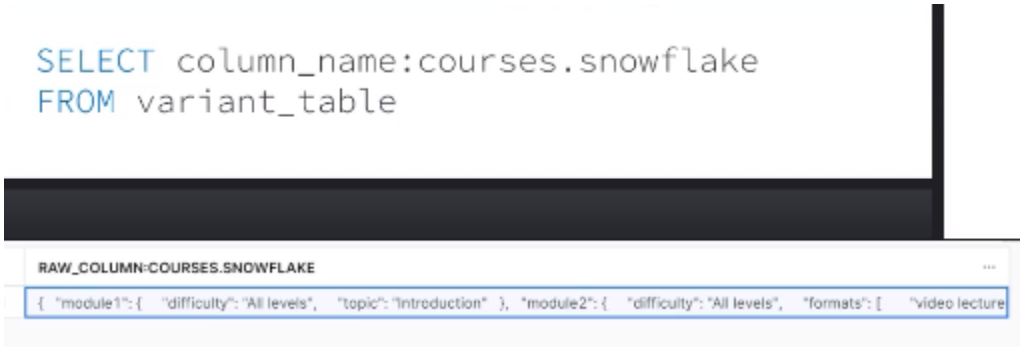

```
{"courses": {
        "snowflake": {
                "module1": {
                                "topic": "Introduction",
                                "difficulty": "All levels"
                                              },
                "module2": {
                                "topic": "Loading data",
                                "formats": [
                                    "video lectures"
                                                ],
                                "difficulty": "All levels"

                                              },
      "azure": {
                "module1": {
                                "topic": "Introduction",
                                "formats": [
                                    "video lectures",
                                    "hands-on",
                                    "quizzes"]
                                              }
                                              }
                                }
                }
```


2. Now if we want to access the courses …

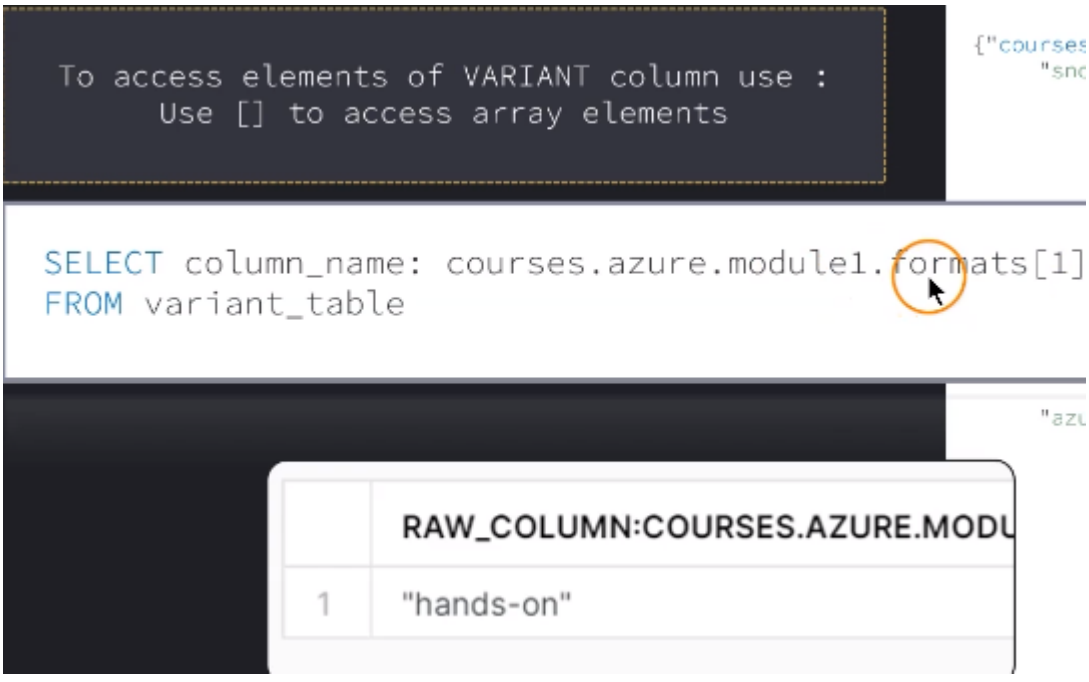


3. We can also use column number..instead of col name

4. If we want to access the snowflake in the variant object..then we have use .



5. Here in azure course…we used arrays for storing formats type..to access the arrays we use



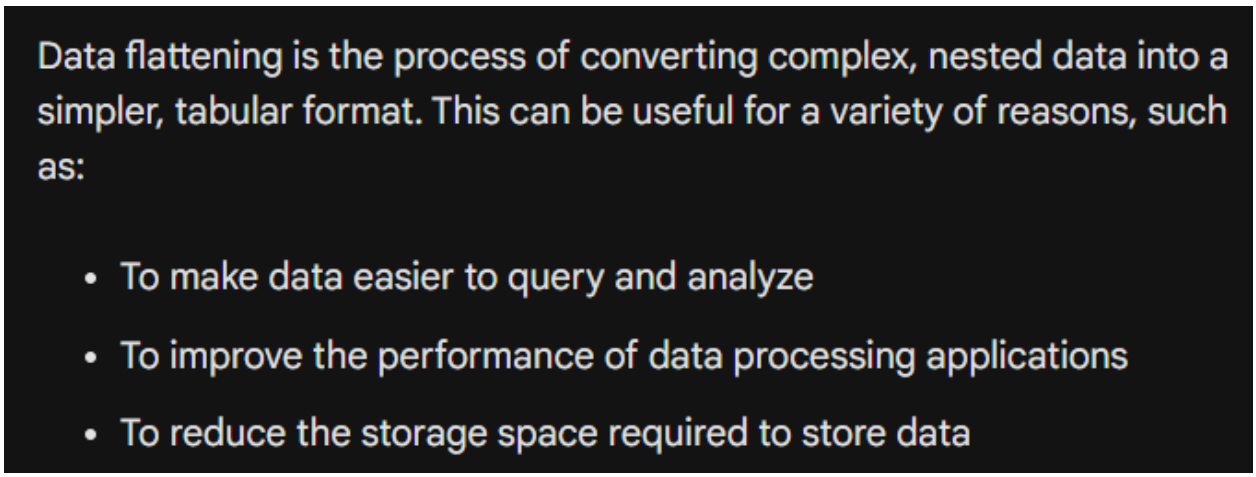
6. If we want to remove semicolons for hands -on we use

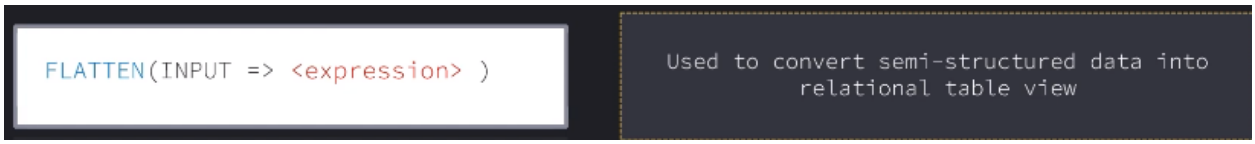
```
SELECT
 column_name: courses.azure.module1.formats[1]::VARCHAR
FROM variant_table
```

HandsOn query semi structured data

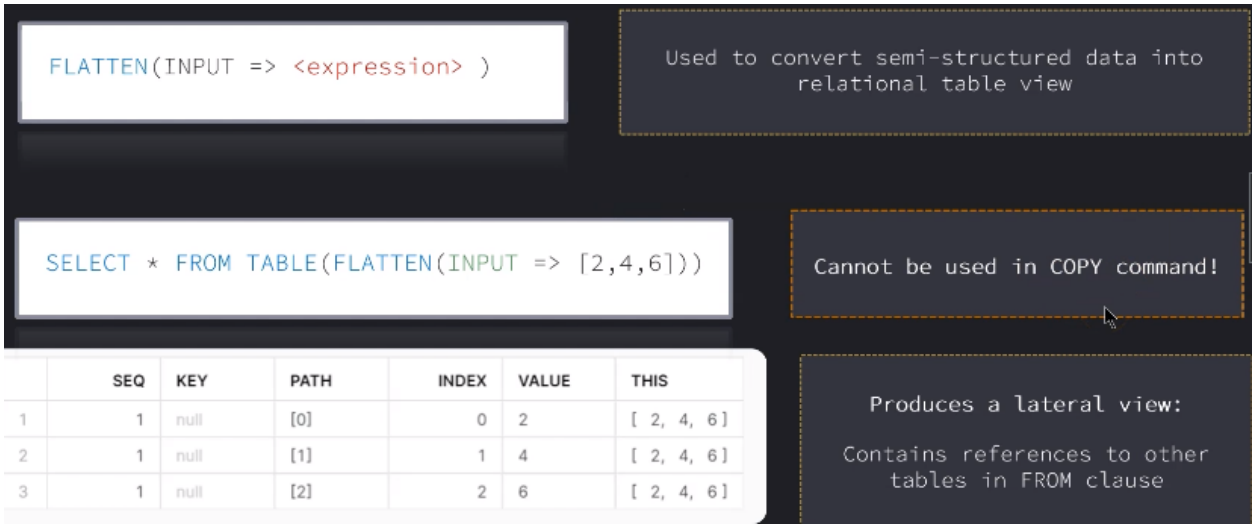
1. Will practice hands on

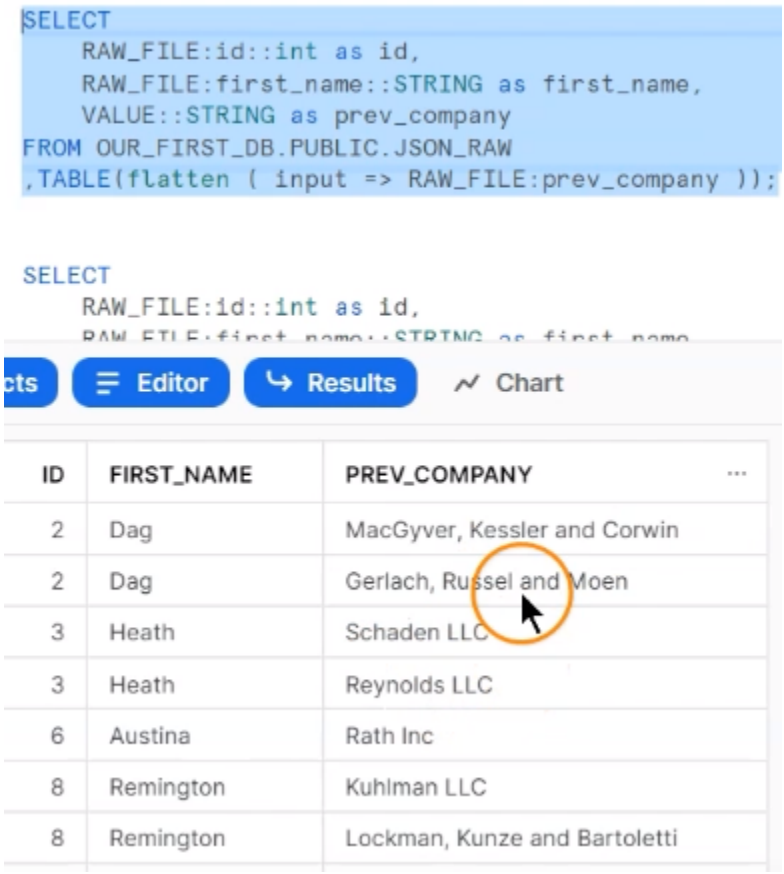
Flatten Hierarchical data

1. 

2. 

3. The flatten function is a table function.
4. This means we can also query from it when we include it in table.

5. 

6. 

Insert JSON Data

1. Lets see how we can manually insert json data in variant

2. First we use parse_join..to convert our data into JSON format

```
SELECT parse_json(' { "key1": "value1", "key2": "value2" } ');
```

3. Now we will create a table with VARIANT data
4. Then we insert

```
CREATE OR REPLACE TABLE semi_structured (data variant);

-- Insert data using PARSE_JSON
INSERT INTO semi_structured SELECT parse_json(' { "key1": "value1", "key2": "value2" } ');
```

5.
```
-- Query from table
SELECT DATA:key1 FROM semi_structured
```
It gives output “value1”
6. Now we insert data into our existing table JSON_RAW

```
INSERT INTO OUR_FIRST_DB.PUBLIC.JSON_RAW SELECT (parse_json('
{   "id":9,
    "first_name":"Kelcey",
    "last_name":"Pavlenko",
    "gender":"Male",
    "city":"Zhuyeping",
    "job":{"title":"Pharmacist","salary":31100},
    "spoken_languages":["Nepali","English"]},
'));
```

7. Now we query this semi structured data by flatten it

```
SELECT
    RAW_FILE:id::int as id,
    RAW_FILE:first_name::STRING as first_name,
    VALUE::STRING as prev_company
FROM OUR_FIRST_DB.PUBLIC.JSON_RAW
,TABLE(flatten ( input => RAW_FILE:prev_company ));
```

```
SELECT
    RAW_FILE:id::int as id,
    RAW_FILE:first_name::STRING as first_name,
    VALUE::STRING as prev_company
FROM OUR_FIRST_DB.PUBLIC.JSON_RAW
,lateral flatten ( input => RAW_FILE:prev_company ));
```

8.

9. Flatten function

The FLATTEN function in Snowflake is a table function that explodes (converts) nested VARIANT, OBJECT, and ARRAY data into rows. It can be used to convert semi-structured data to a relational representation, which makes it easier to query and analyze.

Here is an example of how to use the FLATTEN function to flatten an ARRAY column:

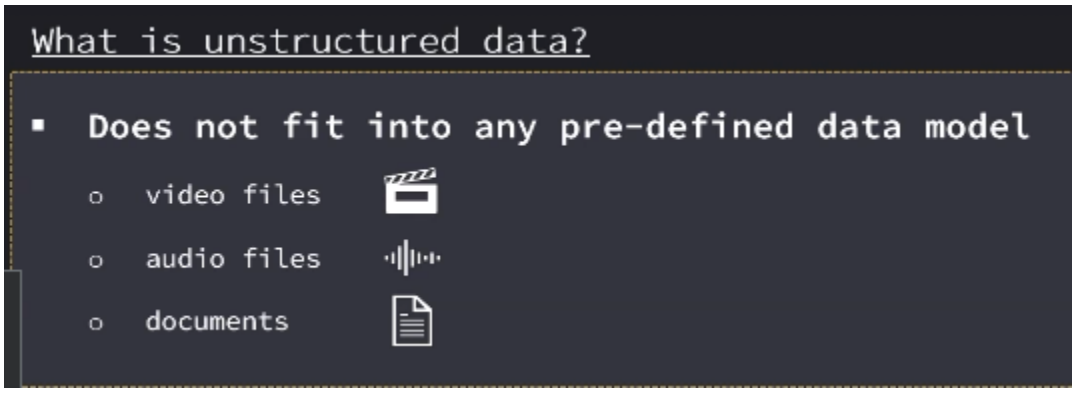
SQL

```sql
CREATE OR REPLACE TABLE my_table (
  id INT,
  names ARRAY<STRING>
);

INSERT INTO my_table (id, names) VALUES
  (1, ARRAY['Alice', 'Bob', 'Carol']),
  (2, ARRAY['Dave', 'Eve', 'Frank']);

SELECT id, name
FROM my_table
LATERAL FLATTEN(names) AS t;
```

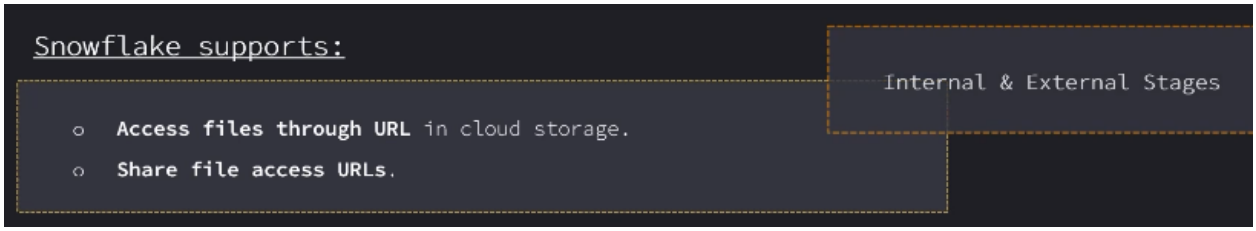
Output:

```
id | name
-- | ----
1  | Alice
1  | Bob
1  | Carol
2  | Dave
2  | Eve
2  | Frank
```

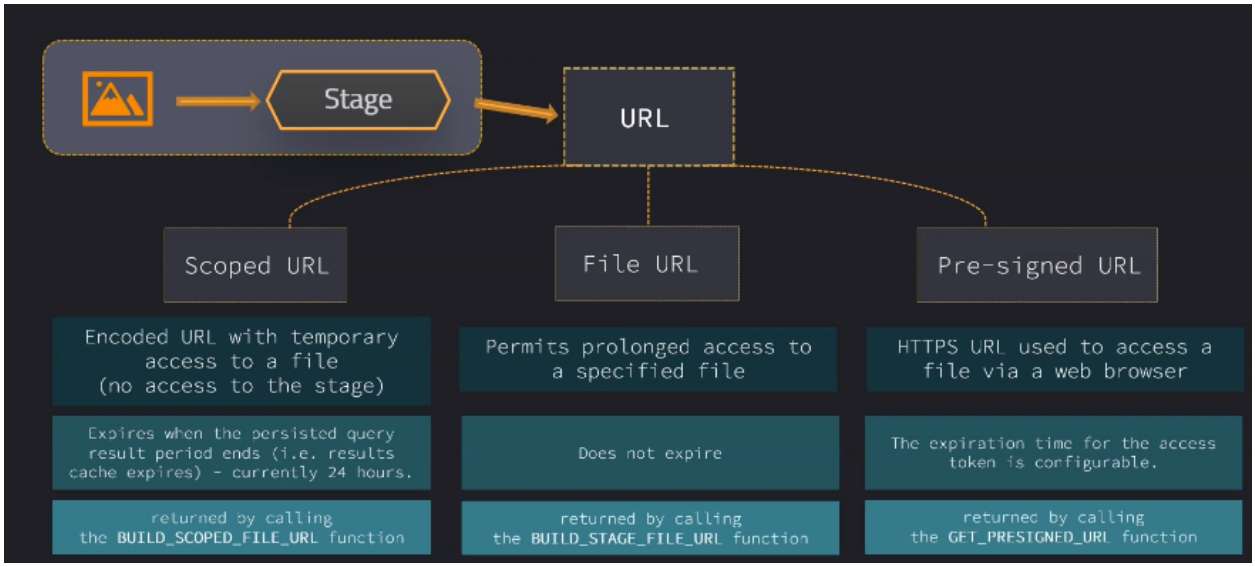
Unstructured Data

1. 

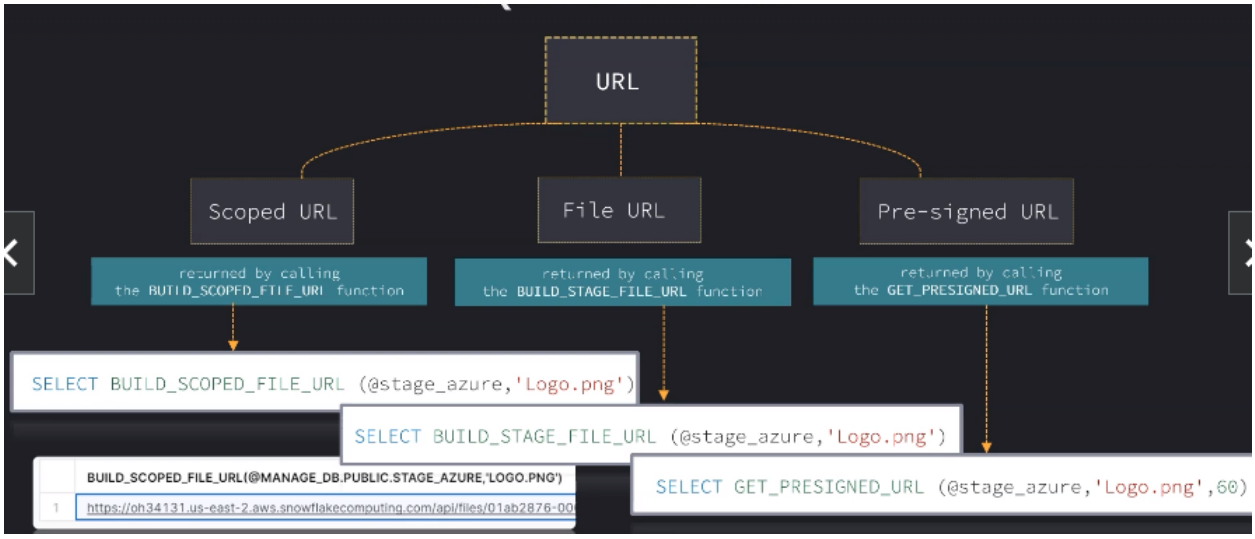
2. And in Snowflake, we can set up URLs through which we can access those files in the cloud storage. And we can also use these URLs to share these files with other people.

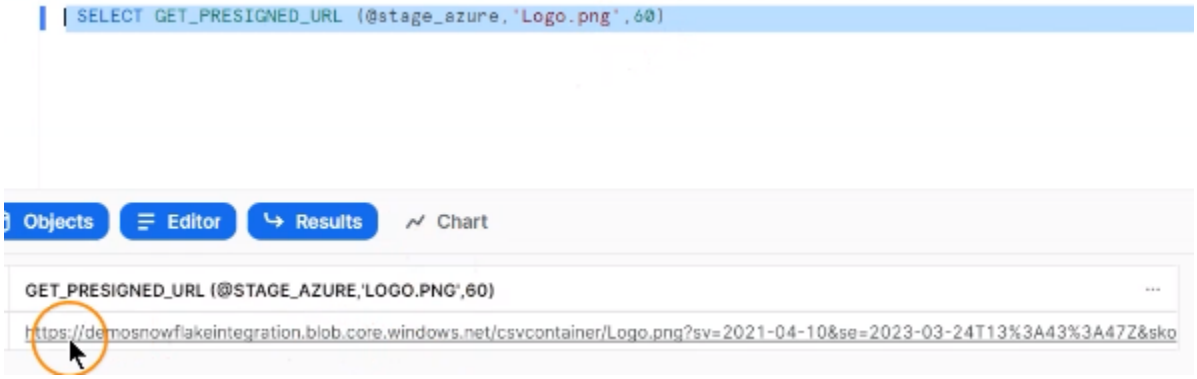


3. Lets suppose we have an image in our stage
4. Now we can generate URLs to access this file and we have 3 types of URLs available



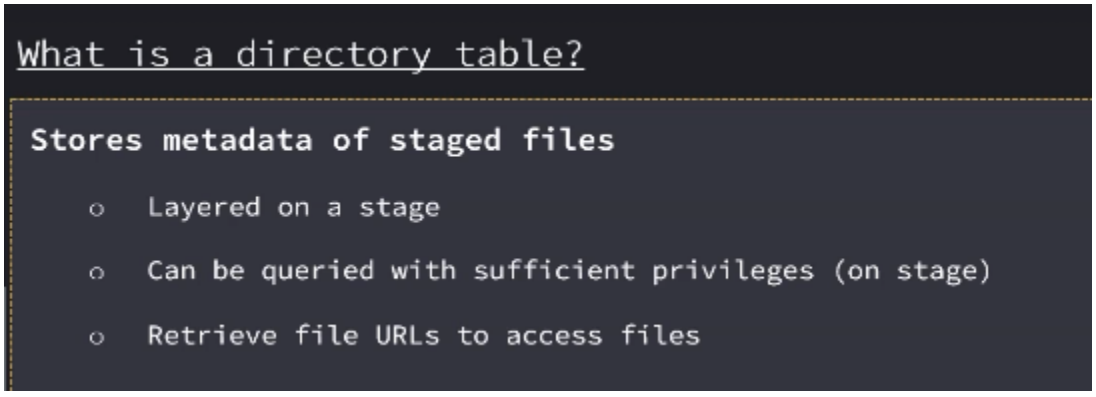
5. TO access them we use
6.
```
SELECT BUILD_SCOPED_FILE_URL (@stage_azure,'Logo.png')
```
at @stage_azure …we have to give path to our file…if it is directly at the stage..then no need of giving it

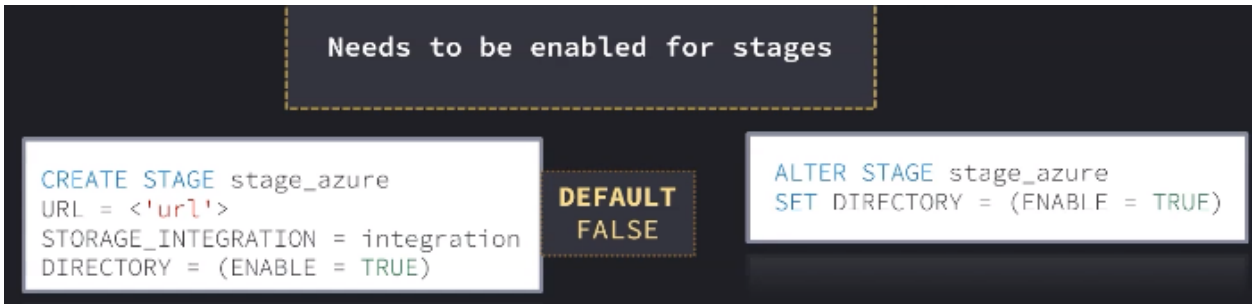
7. 

8. Example


9. Now This url will expire in 60 sec

Directory Tables

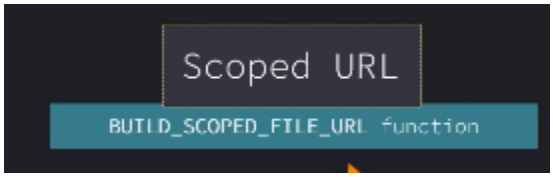
1. 

2. 

3. Now if we run
```
SELECT * FROM DIRECTORY(@stage_azure)
```
4. We get no results…thats because we have to refresh our stage using
```
ALTER STAGE stage_azure REFRESH;
```
5. Now if we query our directory we get

	RELATIVE_PATH	SIZE	LAST_MODIFIED	MD5	ETAG	FILE_URL
	Logo.png	46,603	:19:22.000 -0700	e24ca582347f57f	"0×8DB2C51	https://oh34131.u
2	myfile_0_0_0.csv.gz	33,785	:18:52.000 -0700	a3169775b4e20a	"0×8DB29241	https://oh34131.u

6. Now this file_url is a scoped URL(refer prev lec)



```
1   ALTER STAGE stage_azure
2   SET DIRECTORY = (ENABLE = TRUE)
3
4   SELECT * FROM DIRECTORY( @<stage_name> )
5
6   SELECT * FROM DIRECTORY( @stage_azure )
7
8   ALTER STAGE stage_azure REFRESH;
9
10
11
```

Objects | Editor | Results | Chart

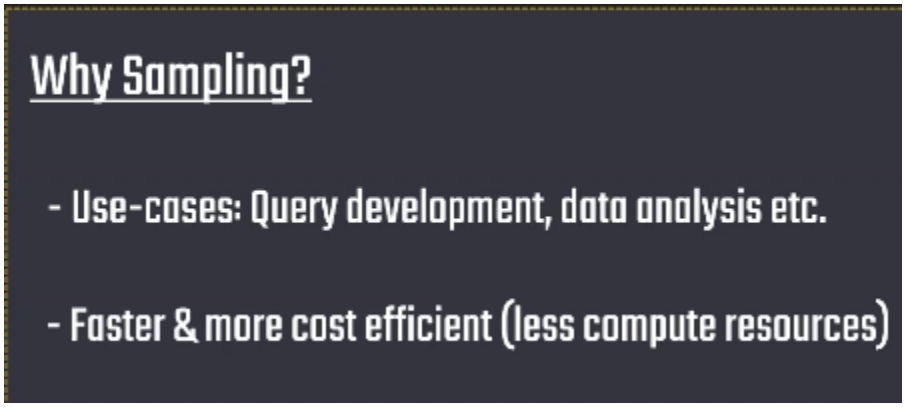
	file	status	description
1	myfile_0_0_0.csv.gz	REGISTERED_NEW	File registered successfully.
2	Logo.png	REGISTERED_NEW	File registered successfully.

7.

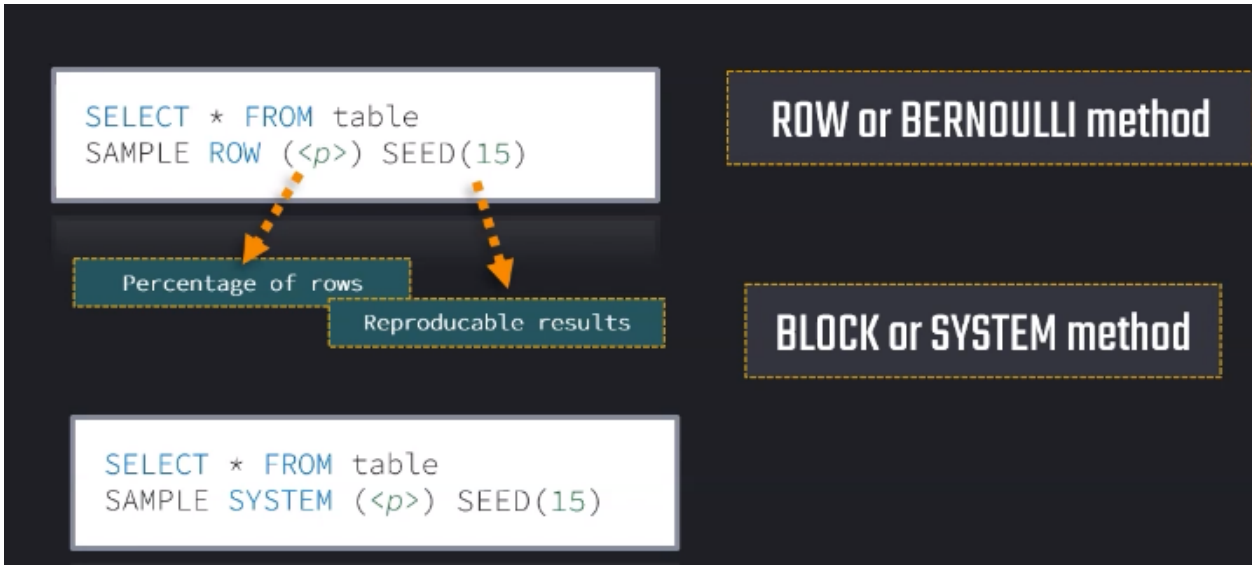
Data Sampling

1. Imagine we have 10TB of data..we need to query some data from this table…
2. Querying from 10tb table will require more time and compute resources
3. So we sample certain amount of data size ..from the table

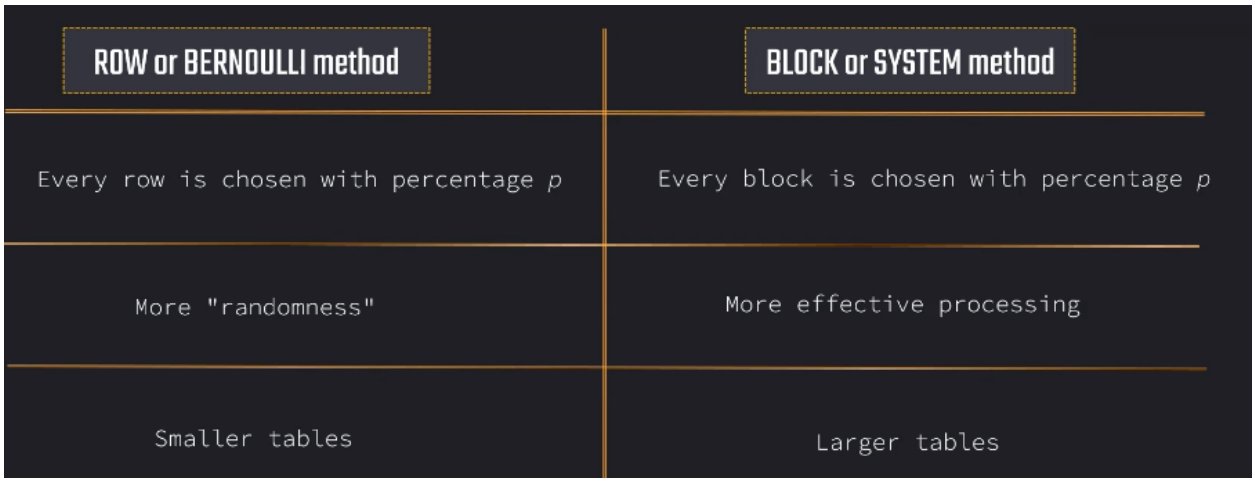
4. It is useful for testing ,data analysis and query dev



5. We have two data sampling method …one is row and next one is Block or System

6. 

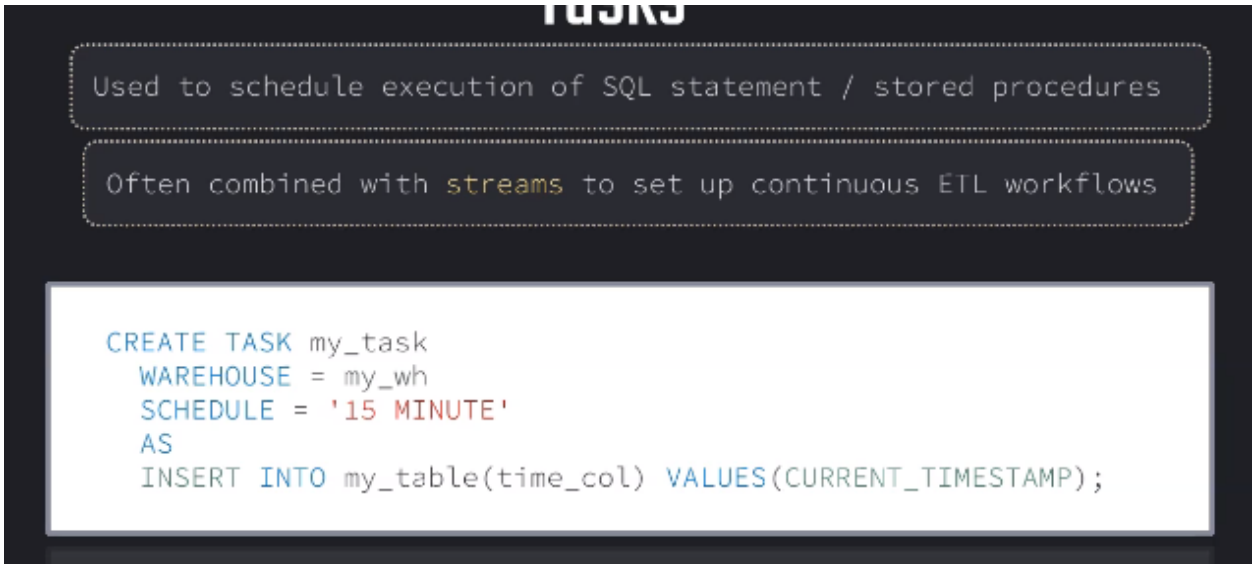
7. Here seed produces reproducible results so that other people can also execute the same query and get exactly the same results.

8. 

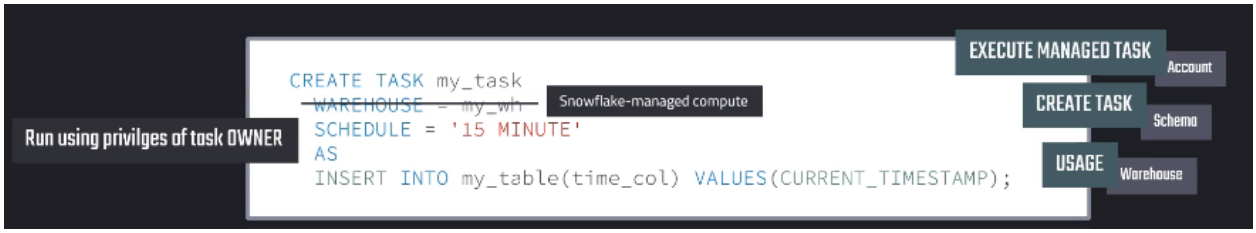
9. In bernoulli ..every row has 10% of chance to be in the sampled data table…in Block…there will be micro partitions of entire table..and it will choose p% from their

10. Bernoulli is more randomness..as it goes to every row in the table …..block is more effective processing..
11. As bernoilli goes to every row..we have to use that in smaller table…and BLOCK for larger tables
12. Once look thru hands on/check online

Tasks

1. 

2. So to setup a task we need to have execute manage task on account level
3. Then we also need to have create task privilege on the schema level and usage on the warehouse.

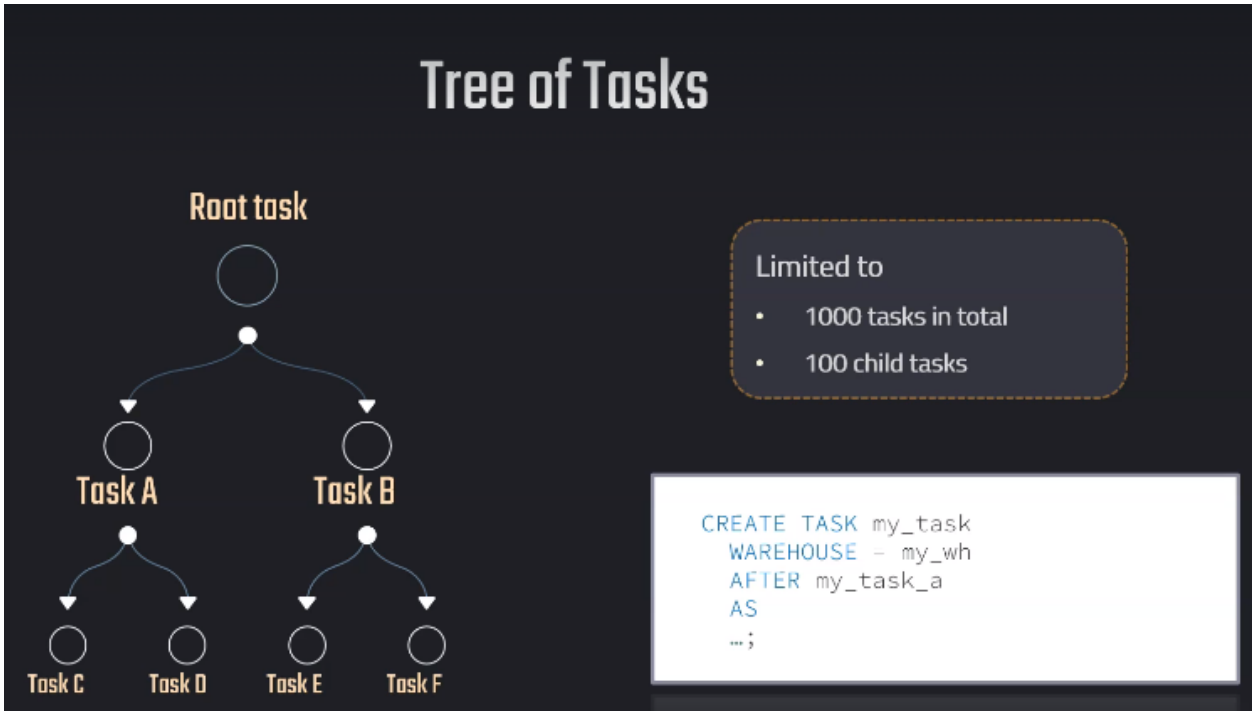


4. If we are not specifying any warehouse..then it will use snowflake managed compute

5. So to start a task ..we need to use resume…and if we want to stop..we use suspend

```
ALTER TASK my_task RESUME;
ALTER TASK my_task SUSPEND;
```

6. We can also have DAG



7. A directed acyclic graph (DAG) in Snowflake tasks is a series of tasks that are organized by their dependencies. A DAG has a single root task, and each task can have multiple child tasks. The child tasks cannot depend on the root task, and there cannot be any cycles in the graph.

   DAGs are useful for automating complex workflows that consist of multiple tasks. For example, you could use a DAG to automate the following workflow:

   1. Load data from a source system into a Snowflake stage.
   2. Transform the data in the stage.
   3. Load the transformed data into a Snowflake table.
   4. Run a machine learning model on the data in the table.
   5. Generate a report based on the results of the machine learning model.

8. Example code for dag

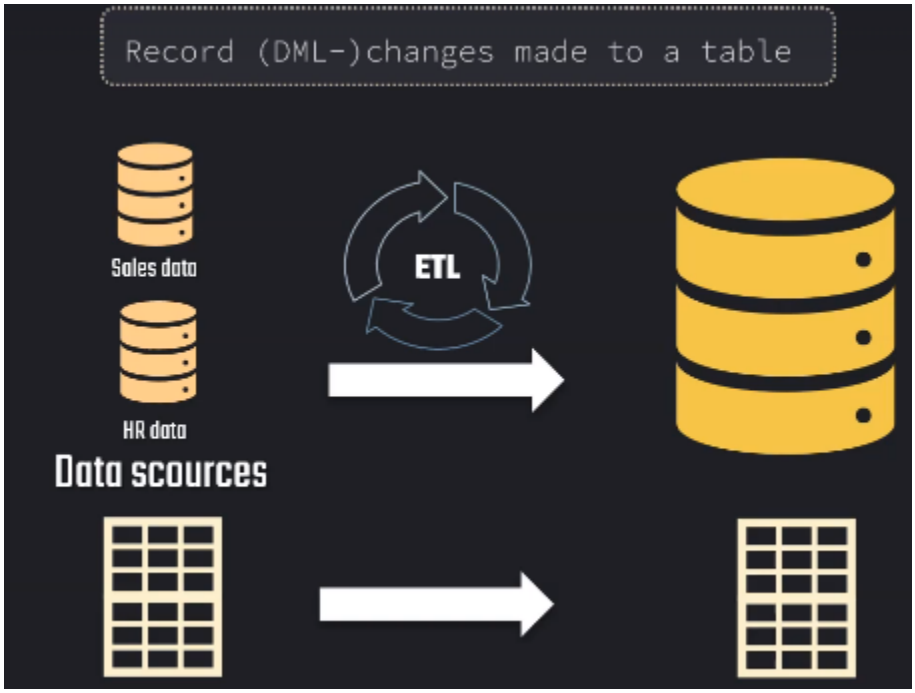
```
CREATE TASK load_data
  WAREHOUSE = my_warehouse
  SCHEMA = my_schema
  STATEMENT = 'COPY INTO my_stage FROM @my_source_system';

CREATE TASK transform_data
  WAREHOUSE = my_warehouse
  SCHEMA = my_schema
  STATEMENT = 'CREATE OR REPLACE TABLE my_table SELECT ... FROM my_stage';

CREATE TASK run_model
  WAREHOUSE = my_warehouse
  SCHEMA = my_schema
  STATEMENT = 'CALL my_model(my_table)';

CREATE TASK generate_report
  WAREHOUSE = my_warehouse
  SCHEMA = my_schema
  STATEMENT = 'CREATE OR REPLACE TABLE my_report SELECT ... FROM my_table';

ALTER TASK load_data
  ADD DEPENDENCY NONE;

ALTER TASK transform_data
  ADD DEPENDENCY load_data;
```

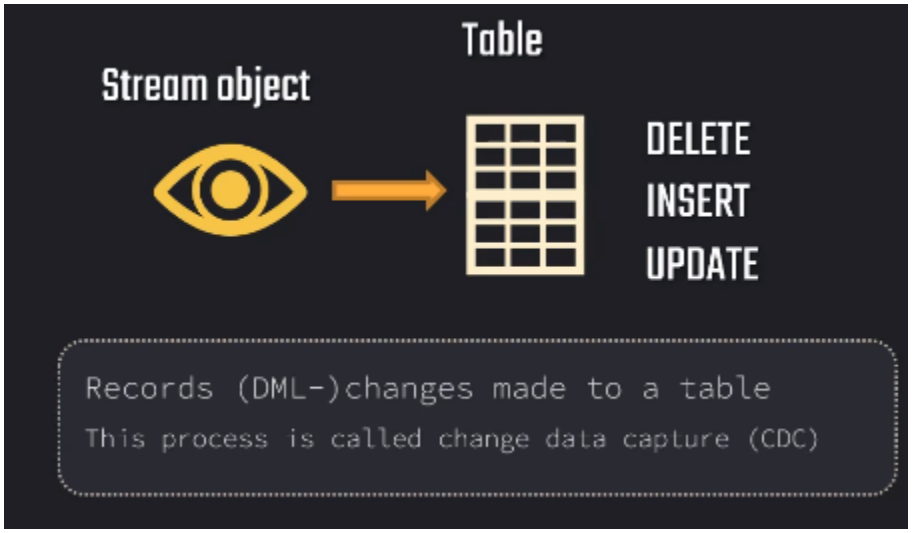
```
ALTER TASK run_model
  ADD DEPENDENCY transform_data;

ALTER TASK generate_report
  ADD DEPENDENCY run_model;
```

Stream

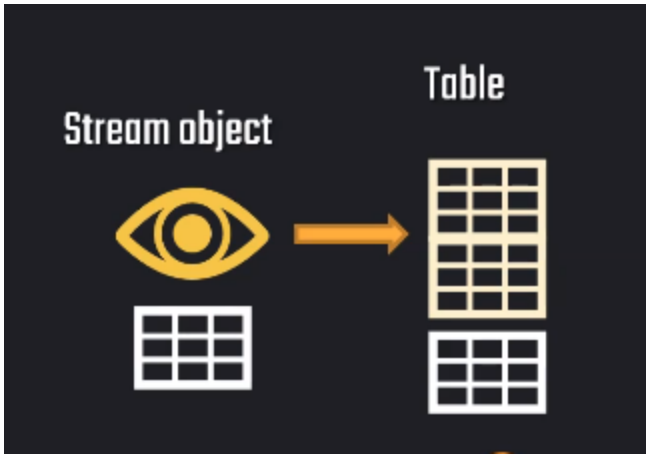
1. A stream is an object that can be used to record data manipulation, language changes.
2. So if there are some inserts, updates or deletes made on a table. A stream can record those changes.

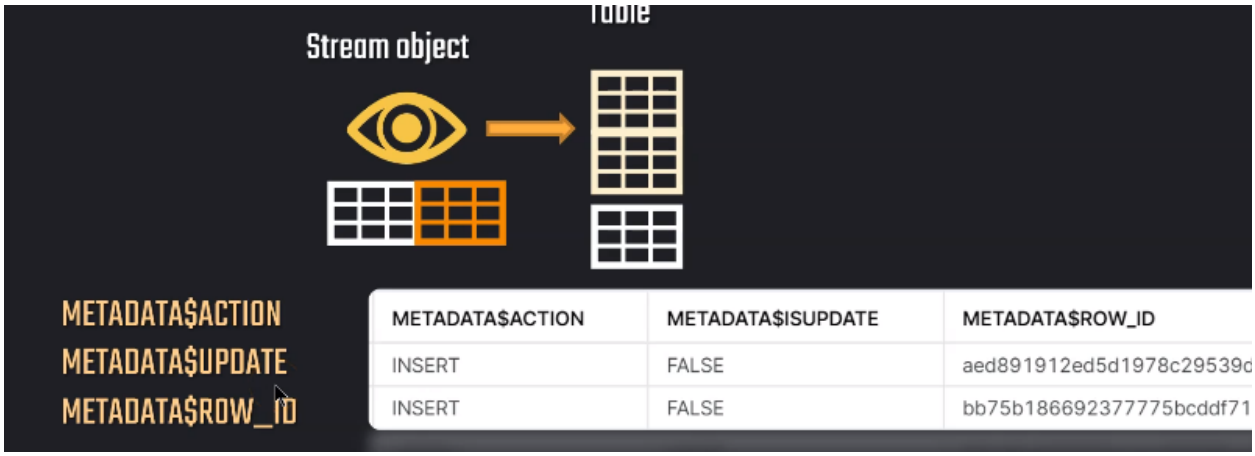
3. It is also useful to track changes…and update these changes in our warehouse



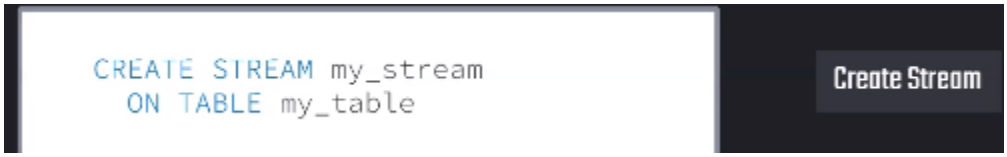
4. 

5. For example…if theres any new rows inserted in our table…stream tracks it and stores the rows that are inserted..and it also stores the meta data





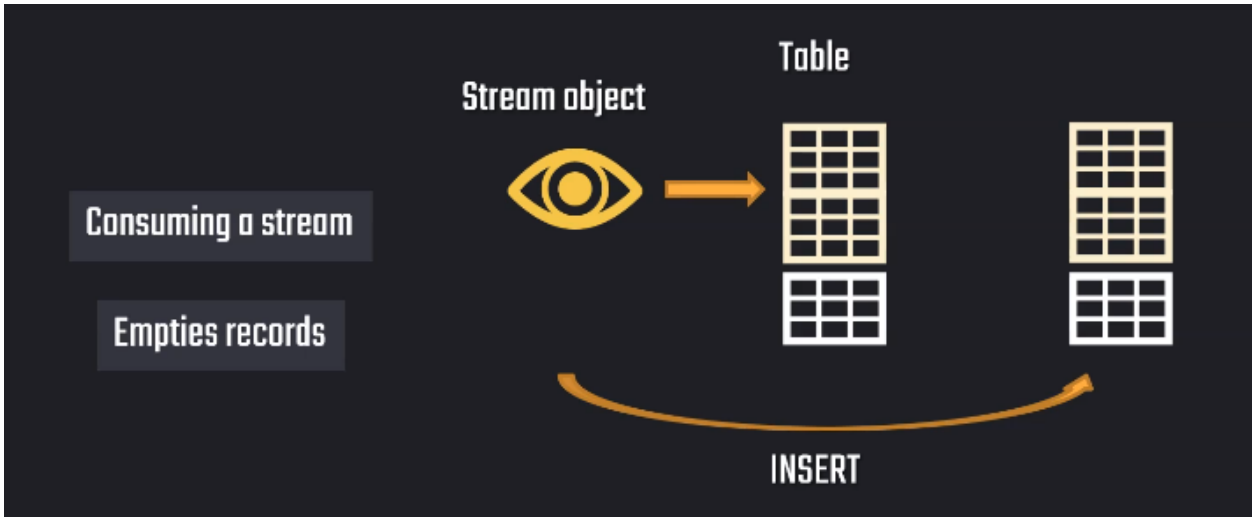
6. To setup a stream we use



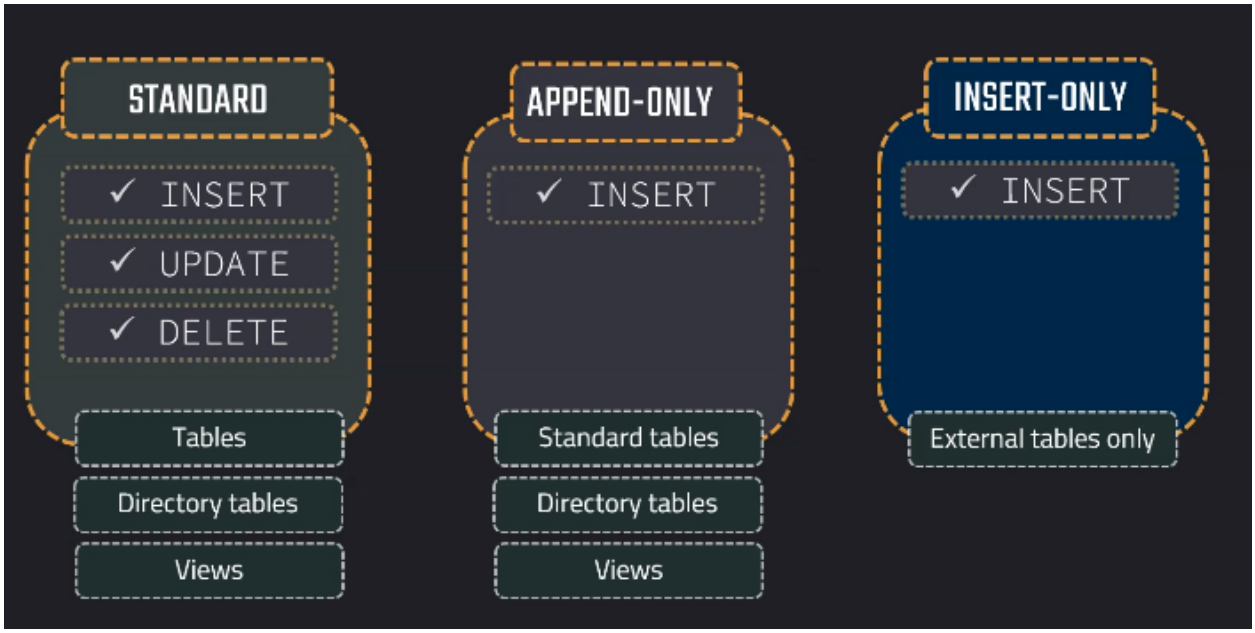
7. Also we can query(new data and metadata) using streams

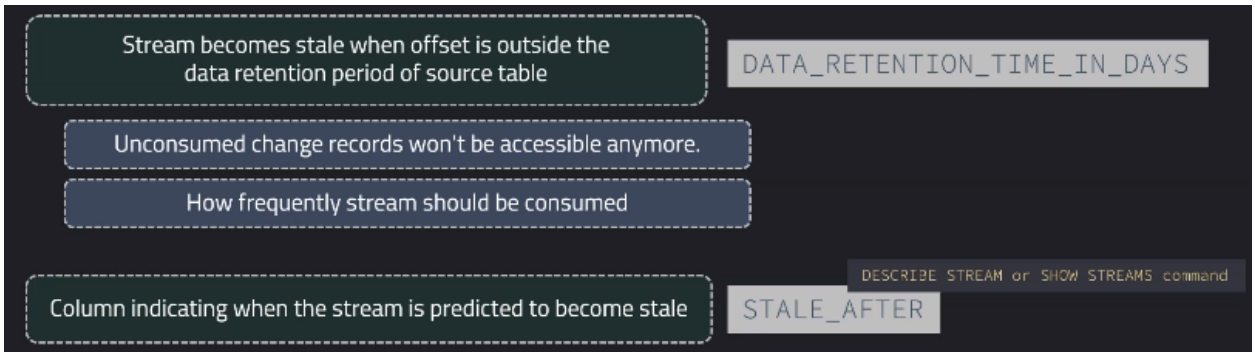
8. In stream…after we insert this newly data to the table..or update the table with existing data…then this will empty the stream object becuz we have consumed

the changes in the stream..and we dont need them anymore

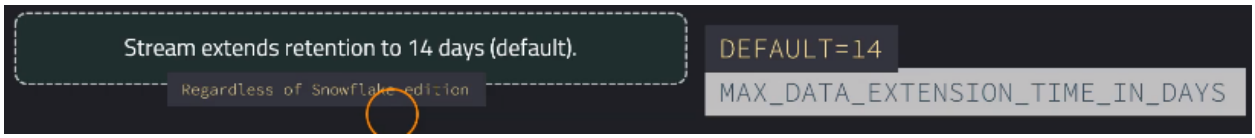


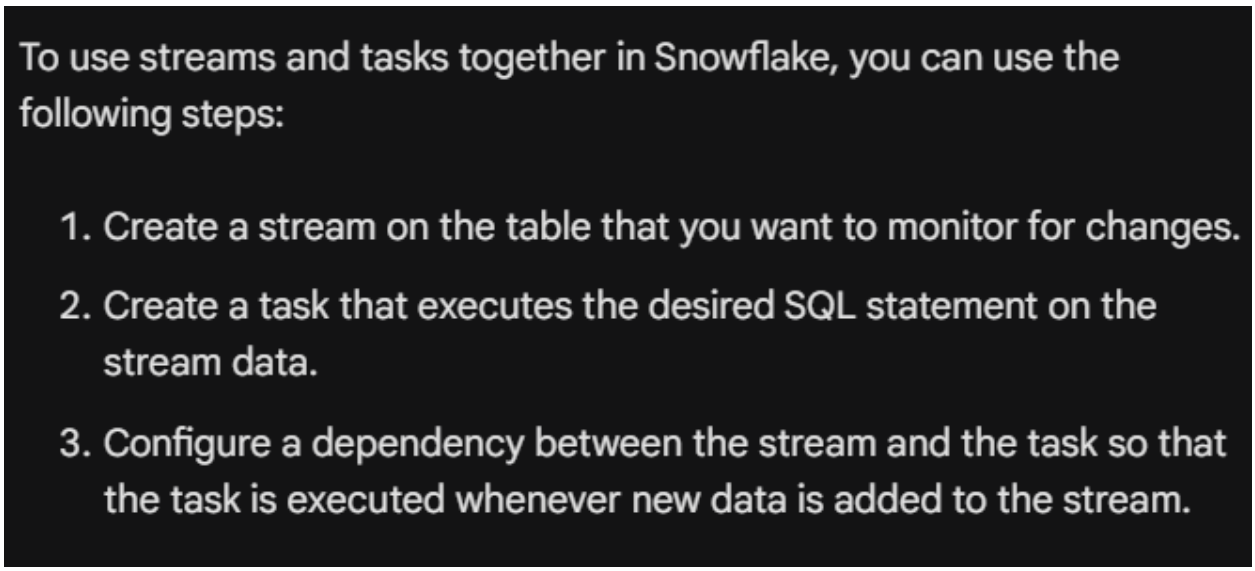
9. So we have 3 types of streams



10. 

11. To check about stream stalness…we use describe stream or show stream command

12. 

13. 


```
CREATE TASK my_task
  WAREHOUSE = my_wh
  SCHEDULE = '15 MINUTE'
  WHEN SYSTEM$STREAM_HAS_DATA('MY_STREAM')
  AS
  INSERT INTO my_table(time_col) VALUES(CURRENT_TIMESTAMP);
```

Handson streams and tasks

1.