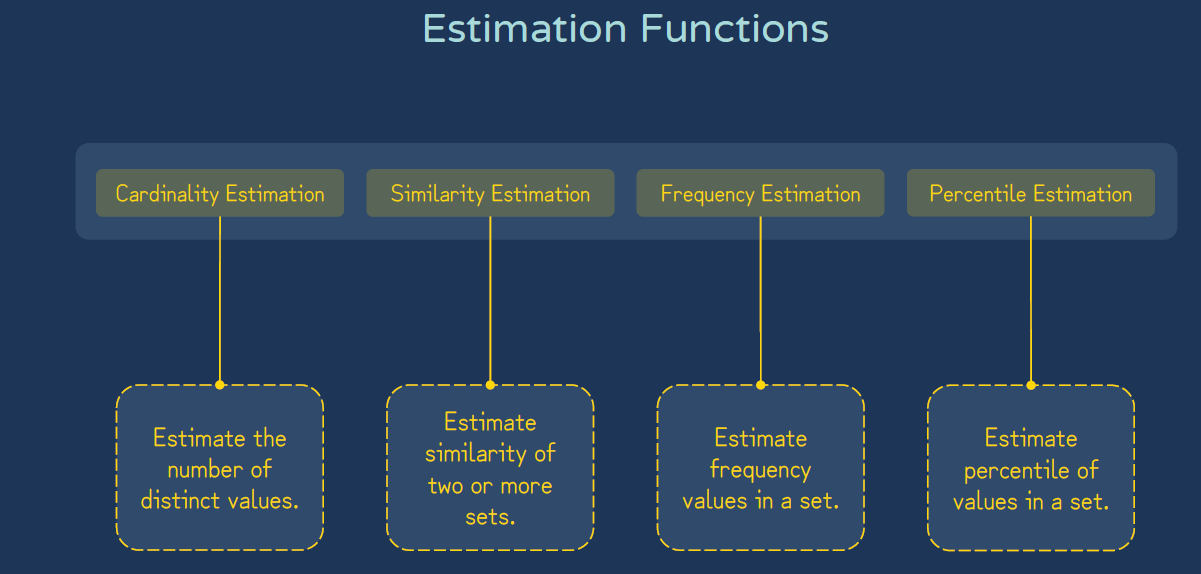
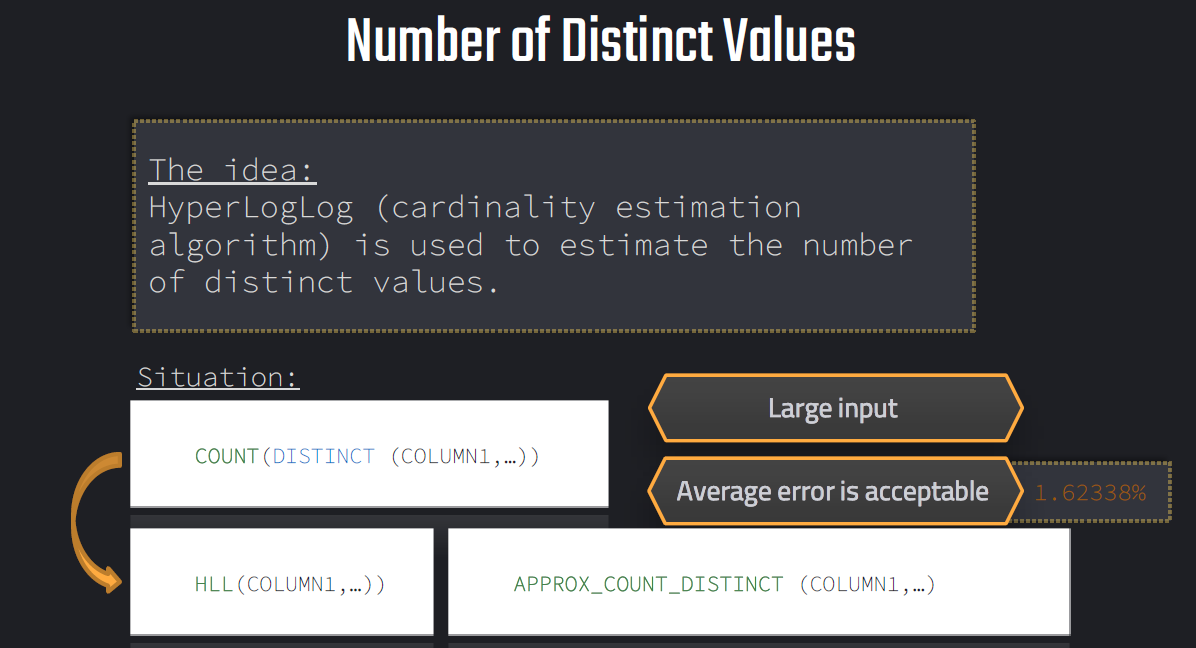
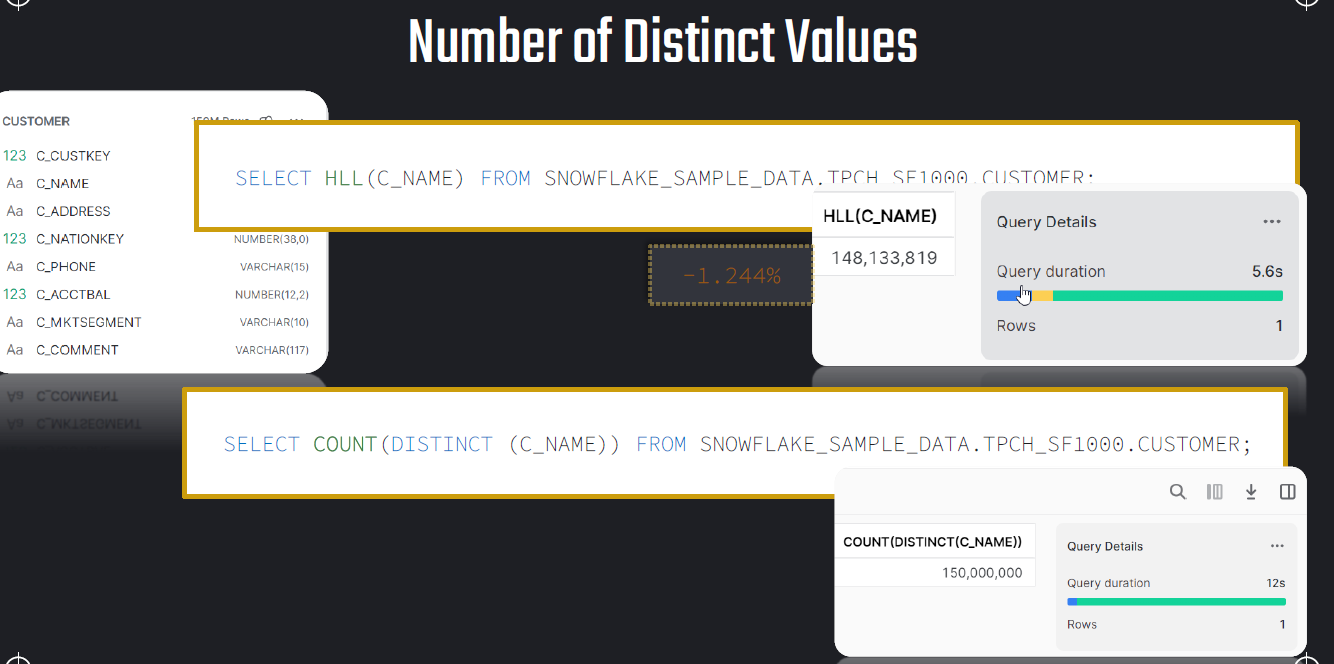


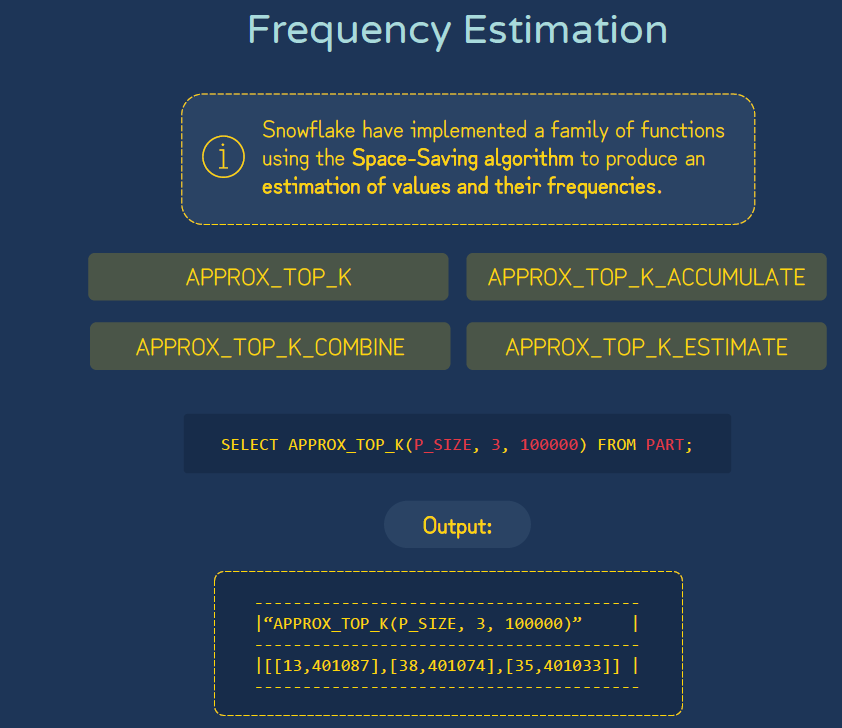
**Estimation functions:**

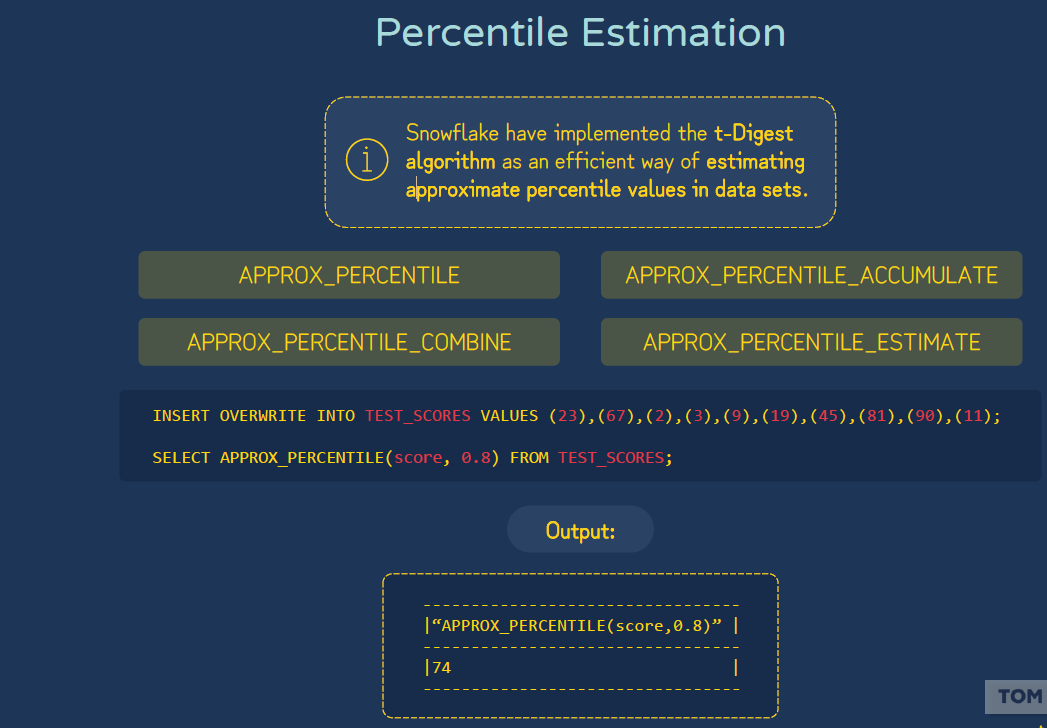
Estimation functions in Snowflake are approximate aggregate functions that use probabilistic algorithms to estimate results such as distinct counts, percentiles, and data distribution metrics.

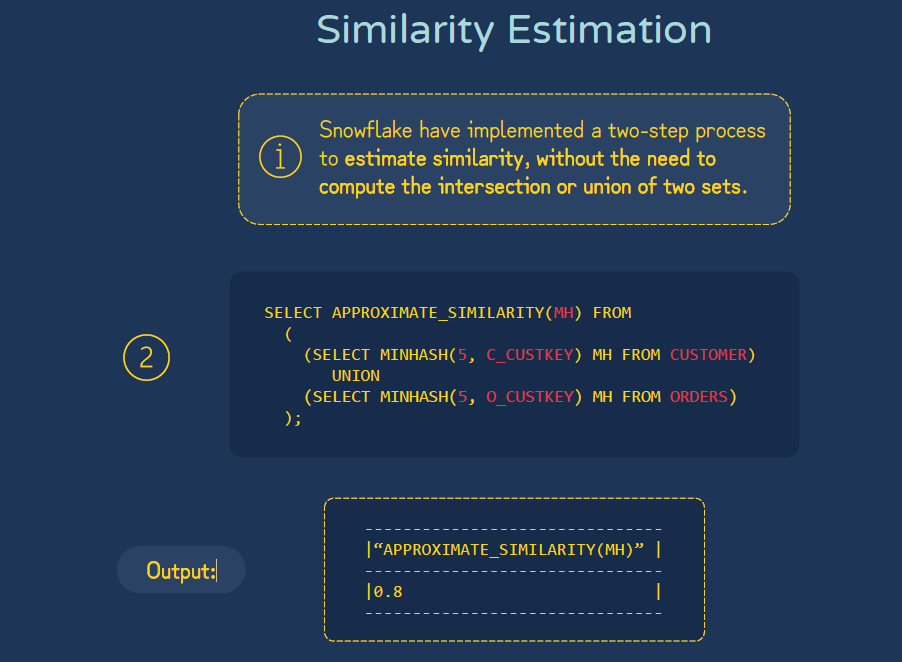












**UDF:**

A User-Defined Function (UDF) allows you to create your own reusable logic that can be called in SQL queries.

| **Type** | **Language** | **Description** |
| --- | --- | --- |
| **SQL UDF** | SQL | Simple functions written in SQL expressions. |
| **JavaScript UDF** | JavaScript | Used for complex logic, loops, or conditionals not supported directly in SQL. |
| **Python UDF (Snowpark)** | Python | Created via Snowpark; allows full Python logic. |
| **Java / Scala UDF** | JVM-based | Used in Snowpark for Java or Scala environments. |

**Example:**

CREATE OR REPLACE FUNCTION ADD\_TWO(N INT)

RETURNS INT

LANGUAGE PYTHON

runtime\_version = '3.9'

handler = 'addtwo'

AS

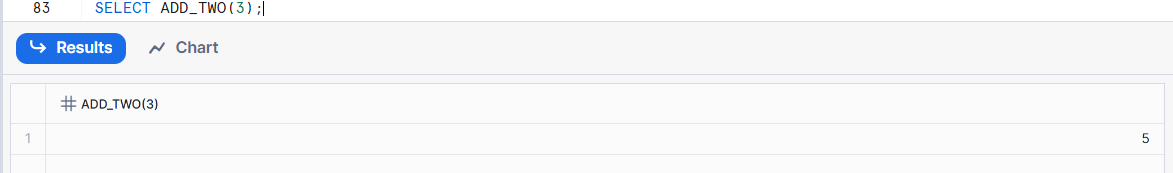
$$

def addtwo(n):

return n+2

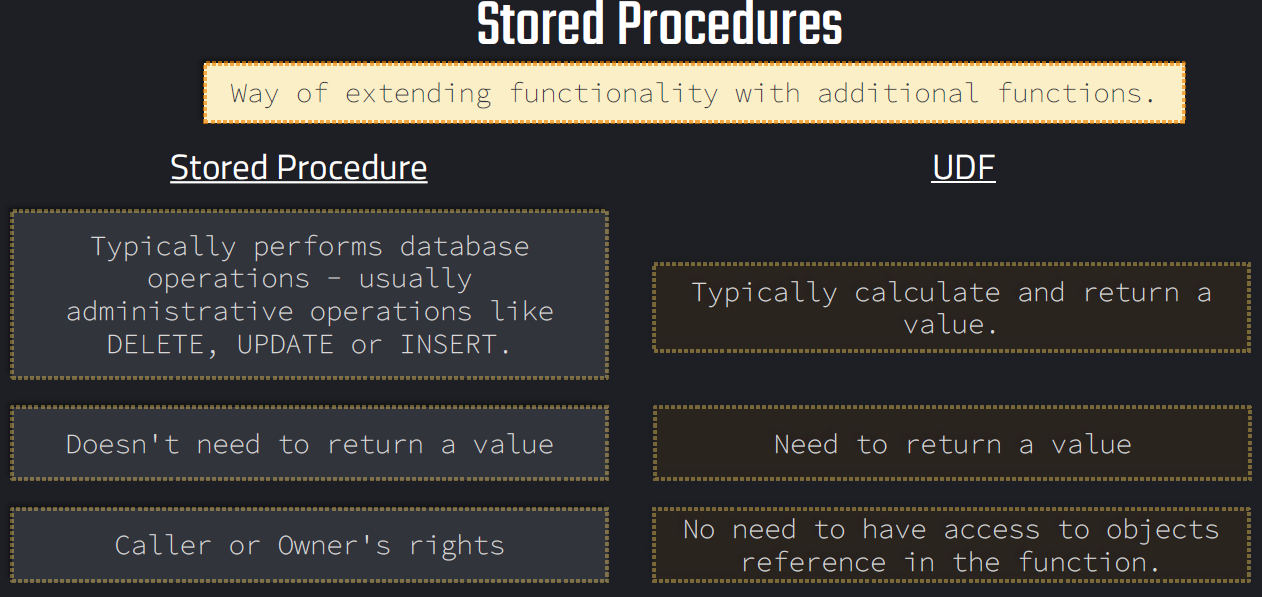
$$;

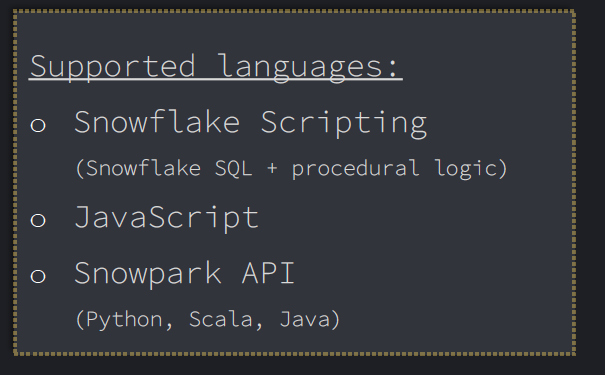
**Calling the function:**

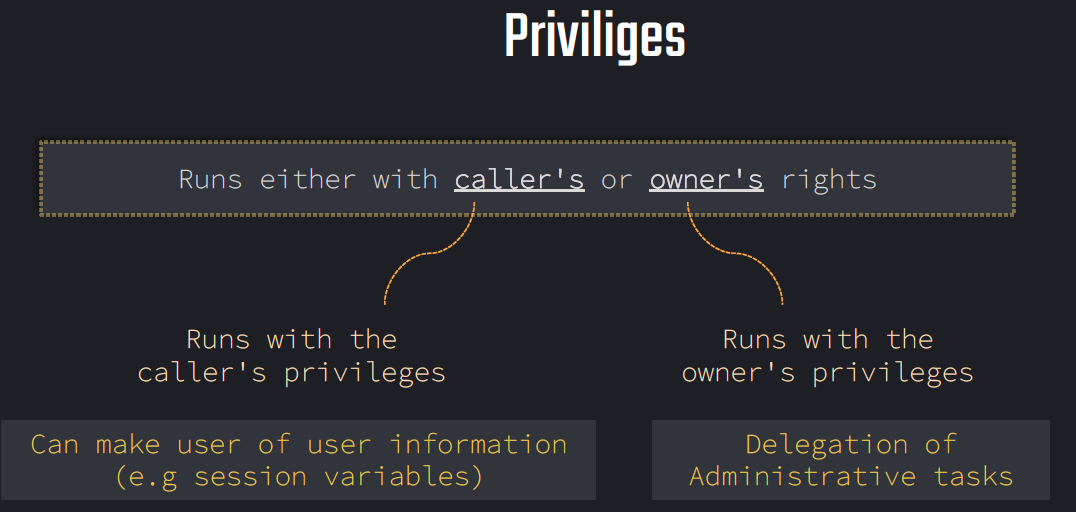
****

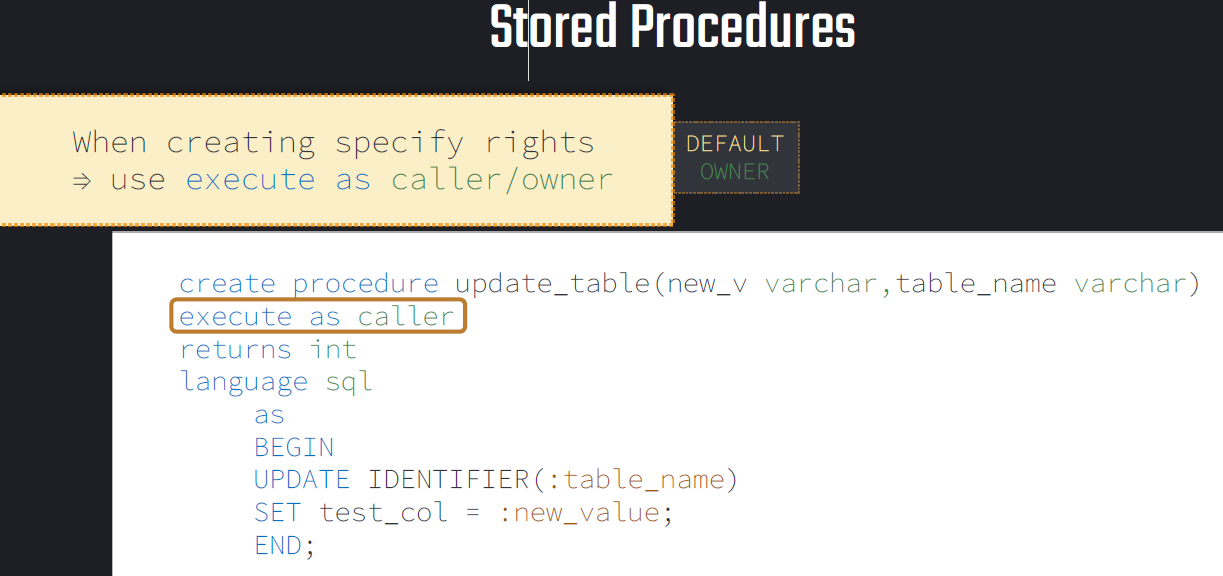
**Stored Procedure:**

Stored Procedure is a programmable object that lets you execute procedural logic — loops, conditionals, dynamic SQL, etc. — inside Snowflake.



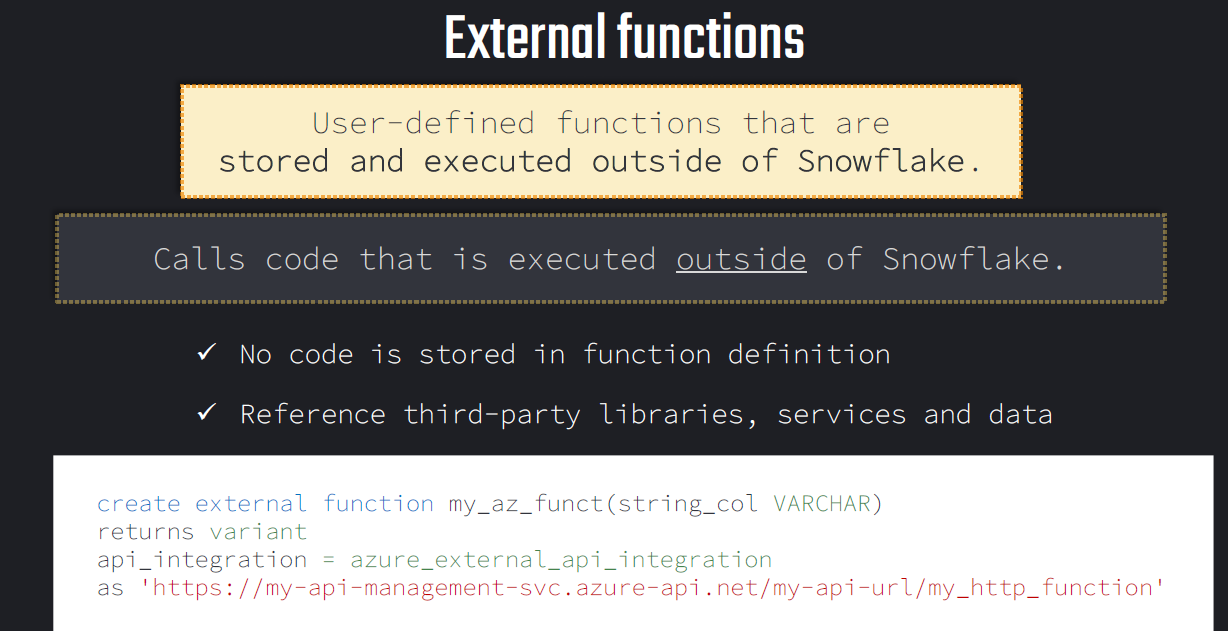


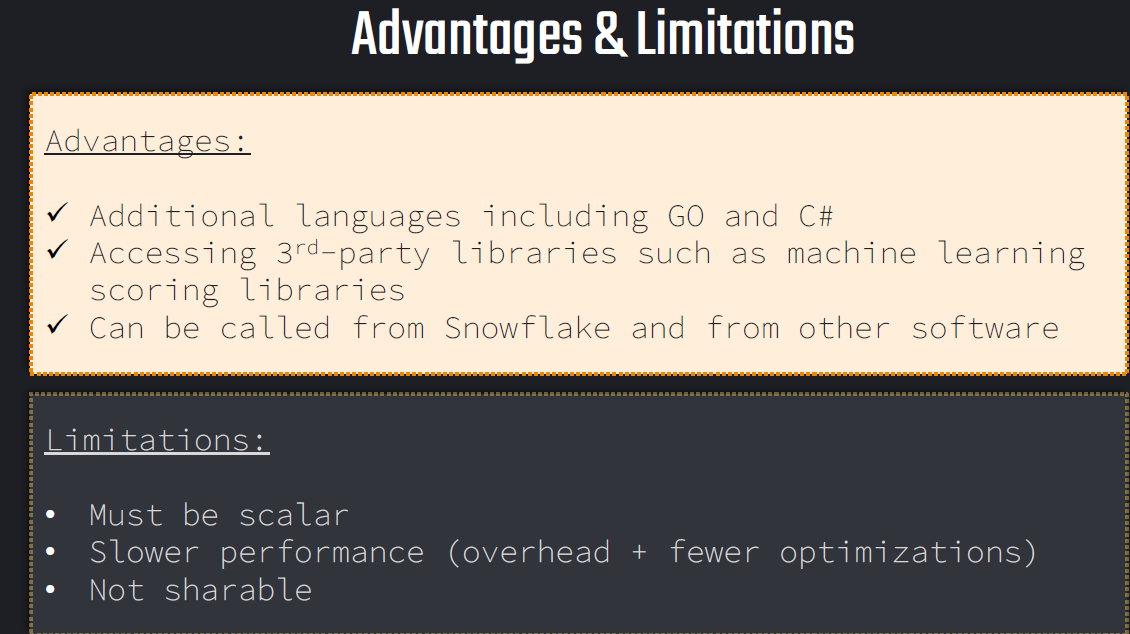


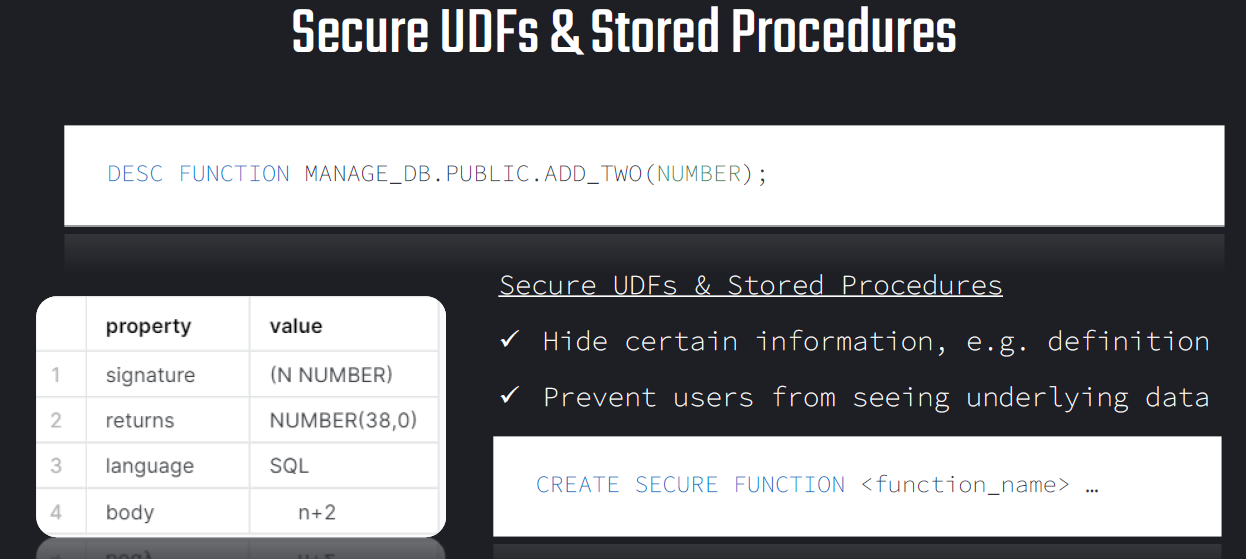


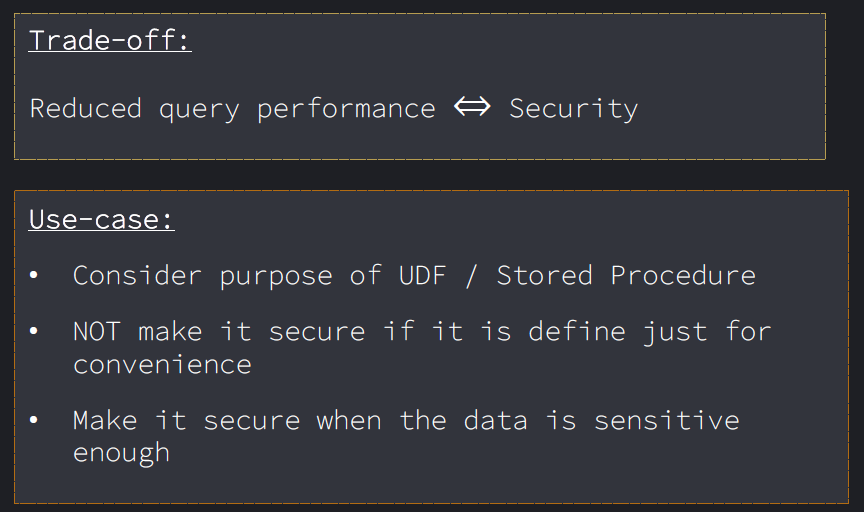
**External Functions:**

They allow you to **call external services or APIs (like AWS Lambda, Azure Functions, GCP Cloud Functions, or REST APIs)** directly from Snowflake SQL — as if they were built-in functions.



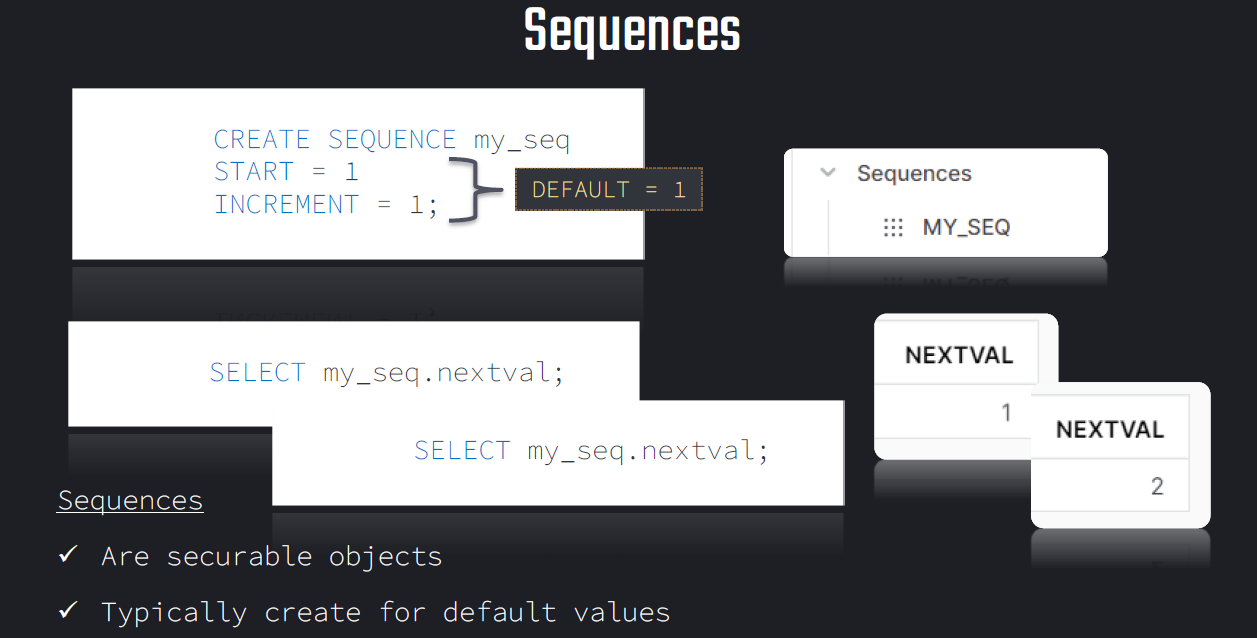


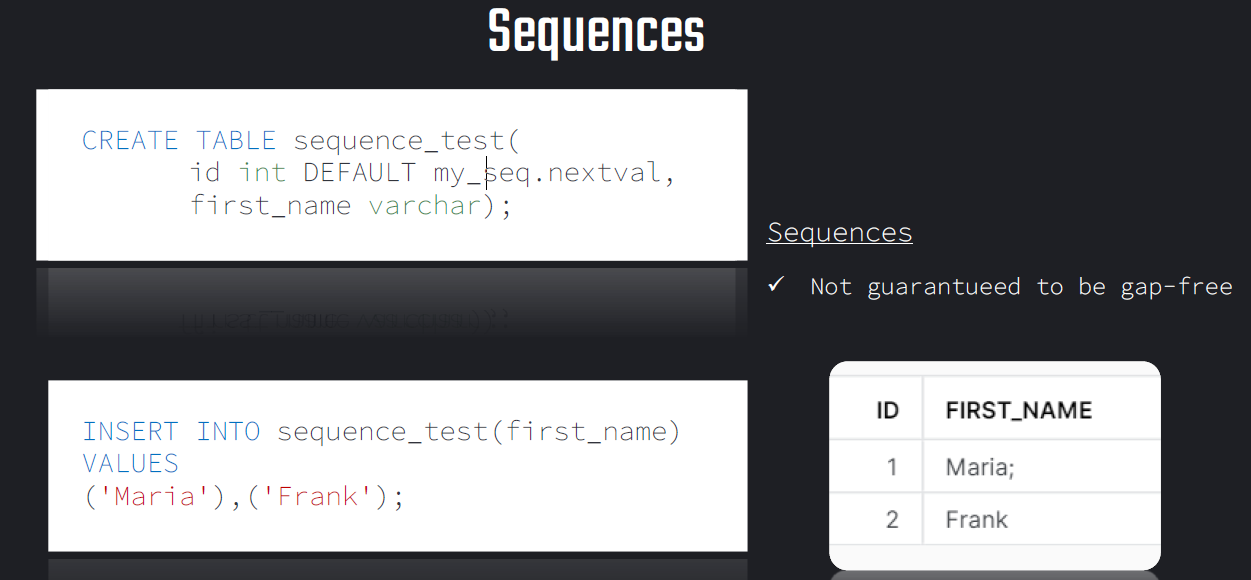


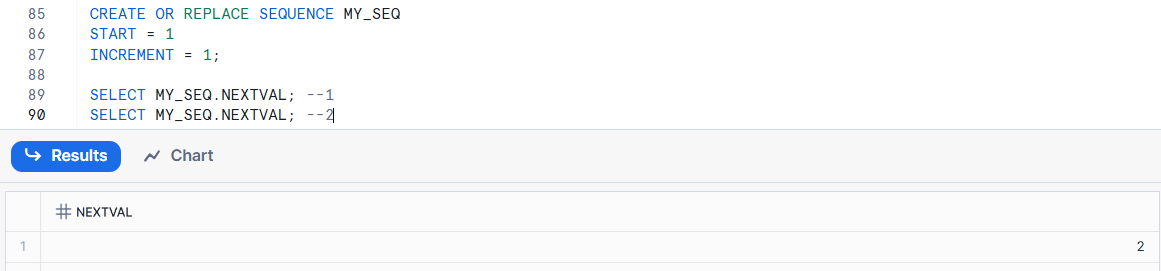


**Sequence:**

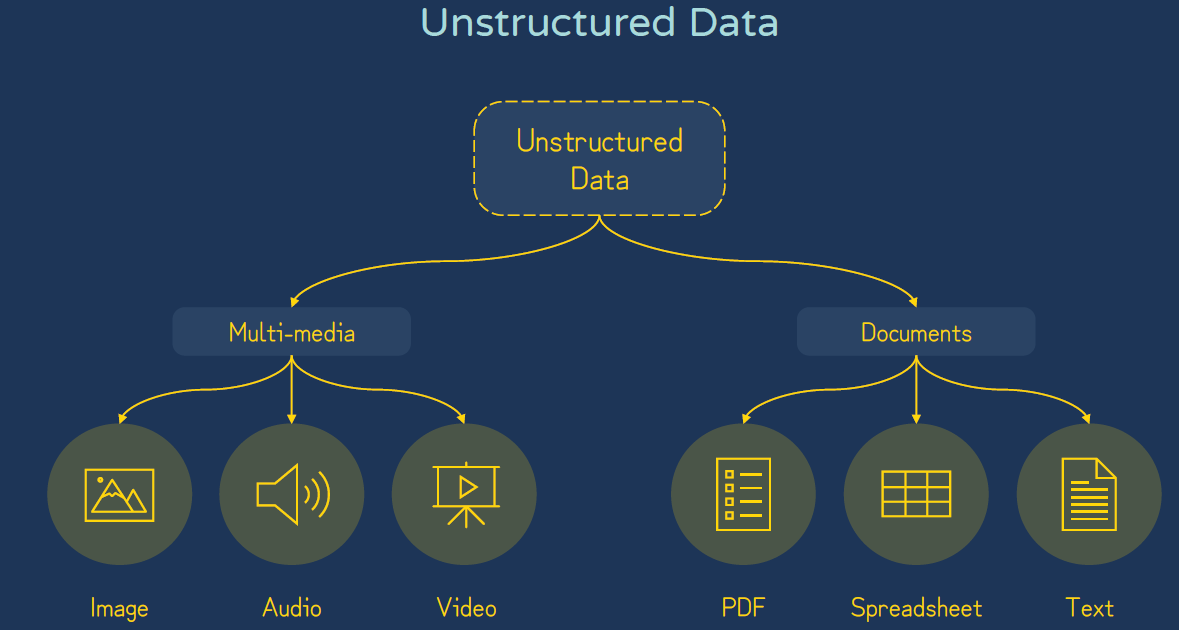
**sequence** is a schema-level object that generates **unique numeric values** — typically used to create **surrogate keys**, **IDs**, or **incremental counters** in tables.

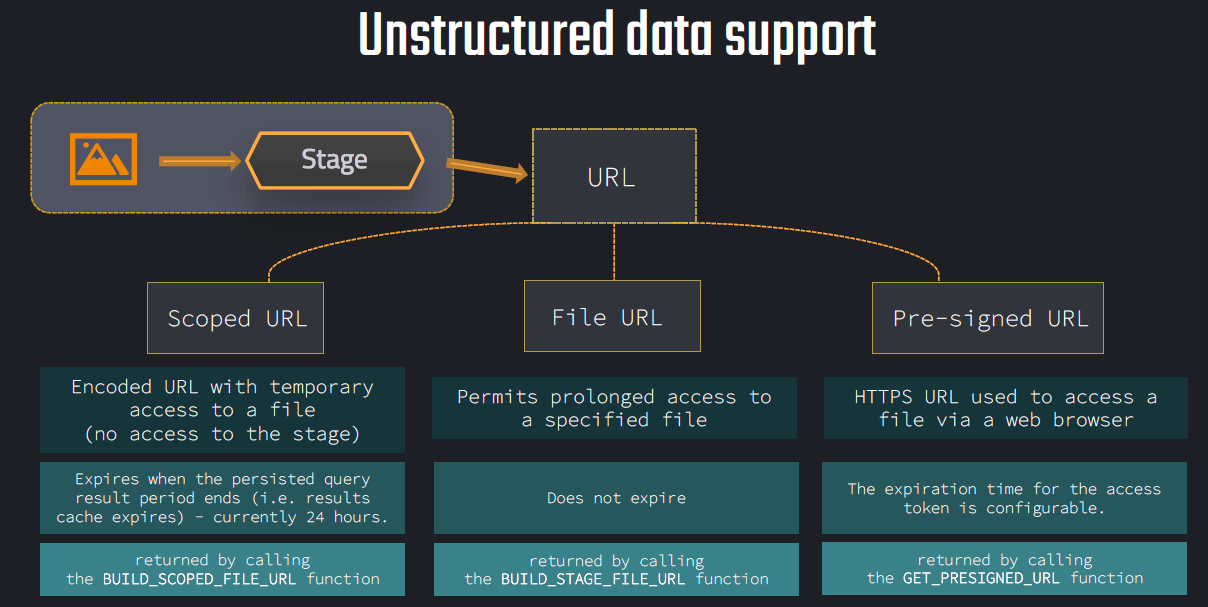


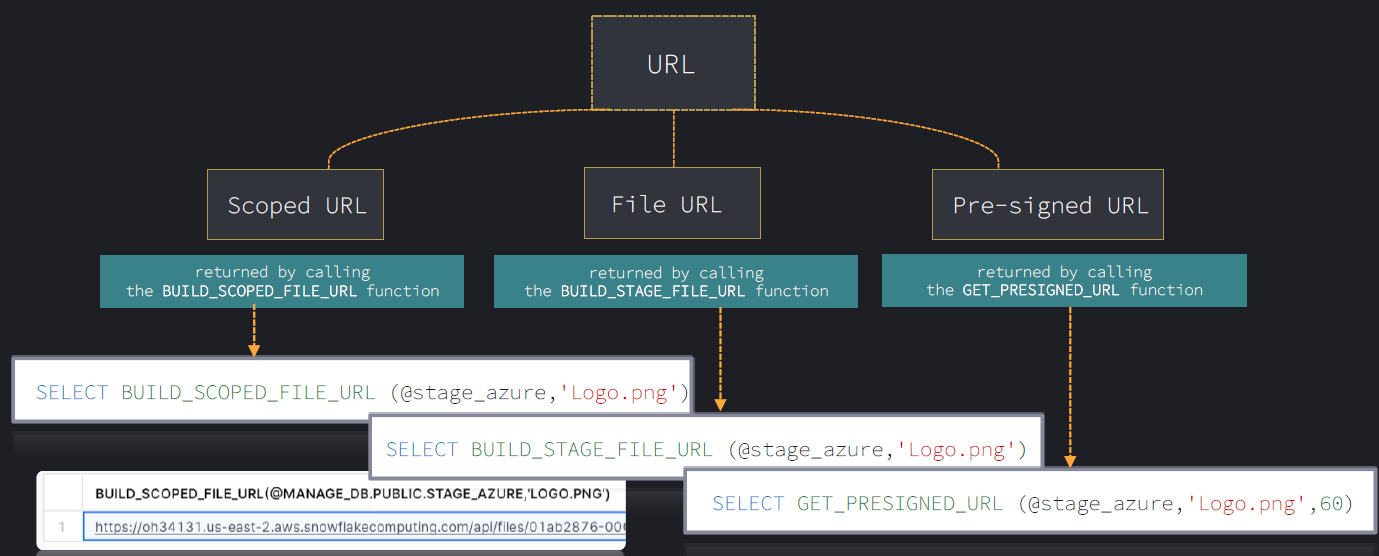


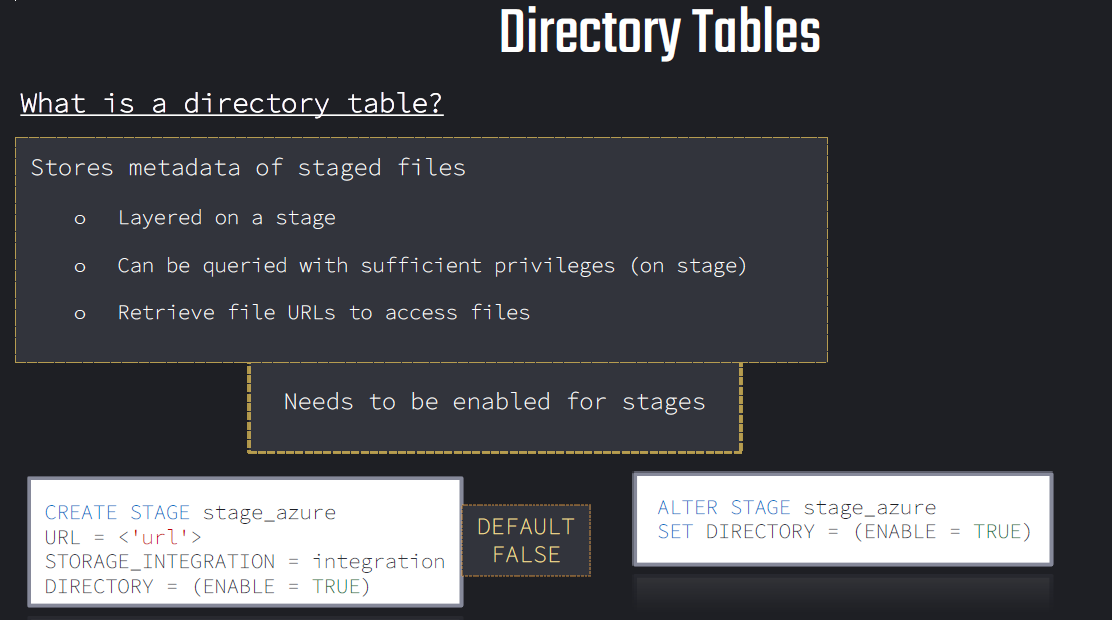


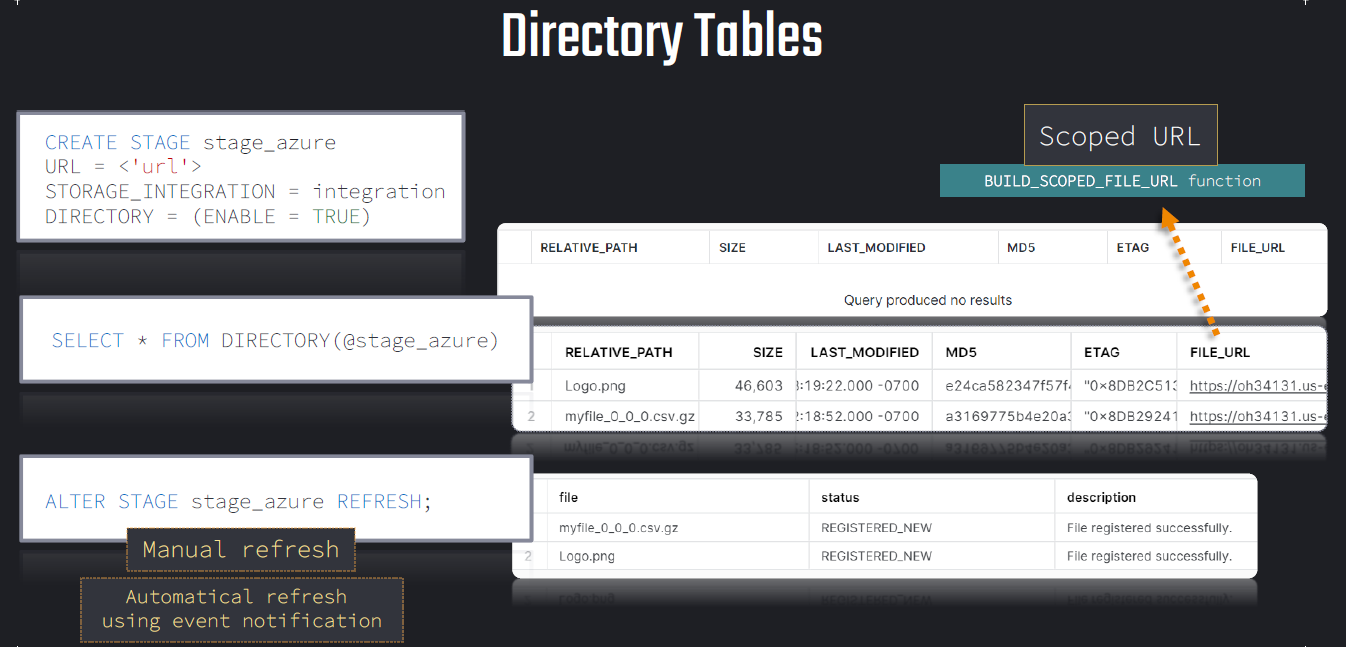
**Unstructured Data:**

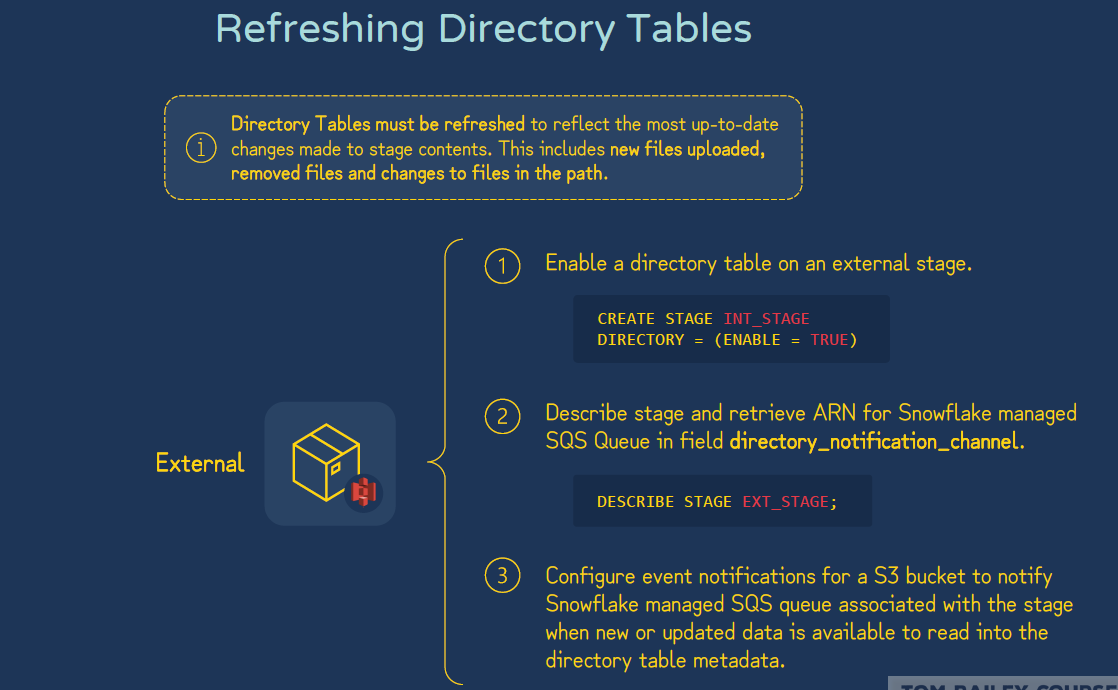
****

****

****

****

****

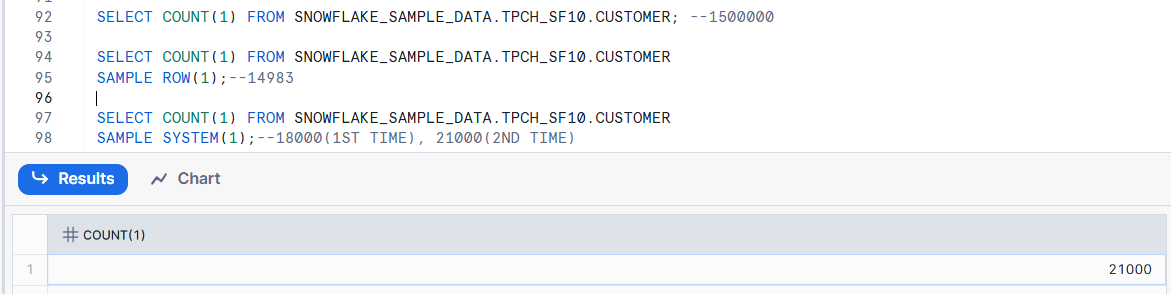
****

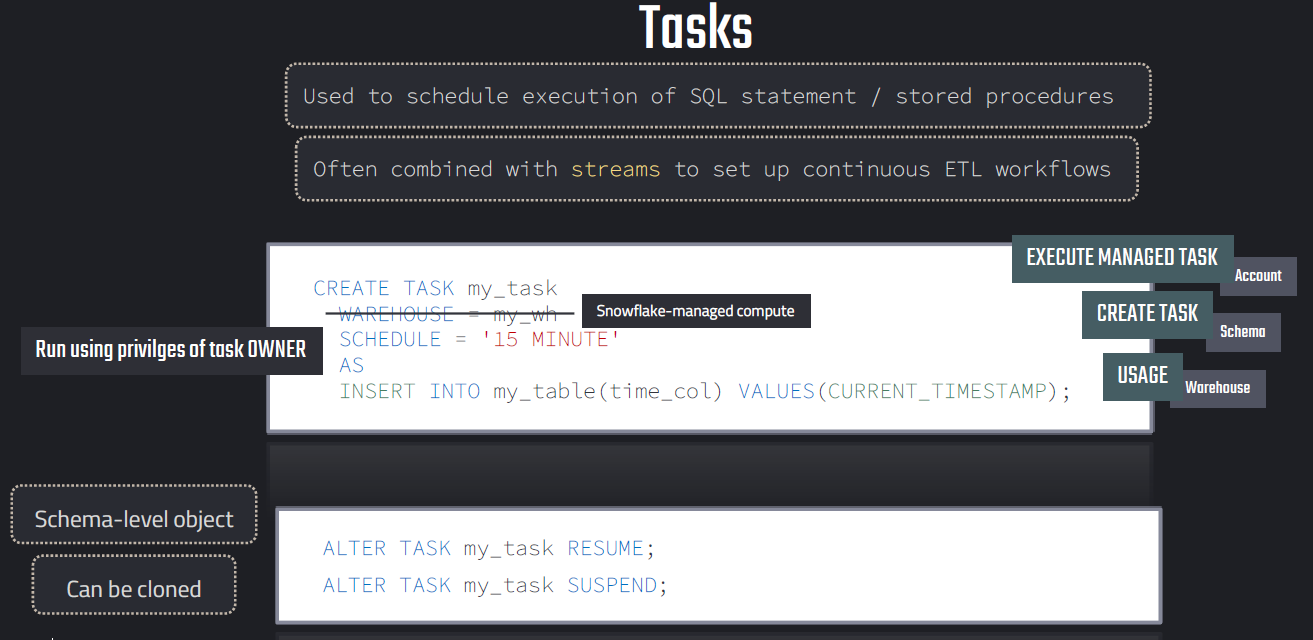
**Data Sampling:**

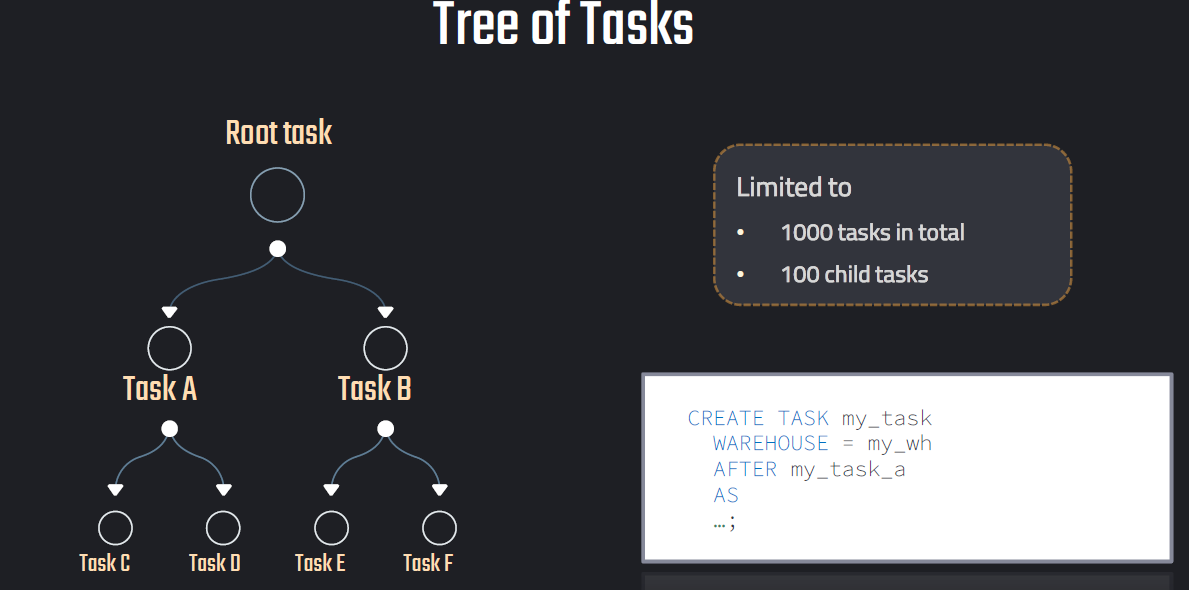
Data Sampling in Snowflake is a technique used to retrieve a subset of rows from a table — typically for testing, data exploration, quality checks, or model training — without scanning the full dataset.

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| **BERNOULLI (or ROW)** | Randomly selects rows from the table. Each row has an independent chance of being included. | SAMPLE BERNOULLI (10) → roughly 10% of rows |
| **SYSTEM (or BLOCK)** | Randomly selects entire micro-partitions (blocks of data). Faster but less granular. | SAMPLE SYSTEM (10) → roughly 10% of micro-partitions |
| **SEED / REPEATABLE** | Produces the same sample each time (deterministic) if you use the same seed value. | SAMPLE (10) REPEATABLE (42) |

| **Feature** | **BERNOULLI** | **SYSTEM** |
| --- | --- | --- |
| Sampling Unit | Rows | Micro-partitions (16MB blocks) |
| Accuracy | More precise (better randomness) | Faster, but less granular |
| Performance | Slower for large data | Faster for large data |
| Use Case | Small to medium tables, testing | Large datasets, performance-sensitive queries |

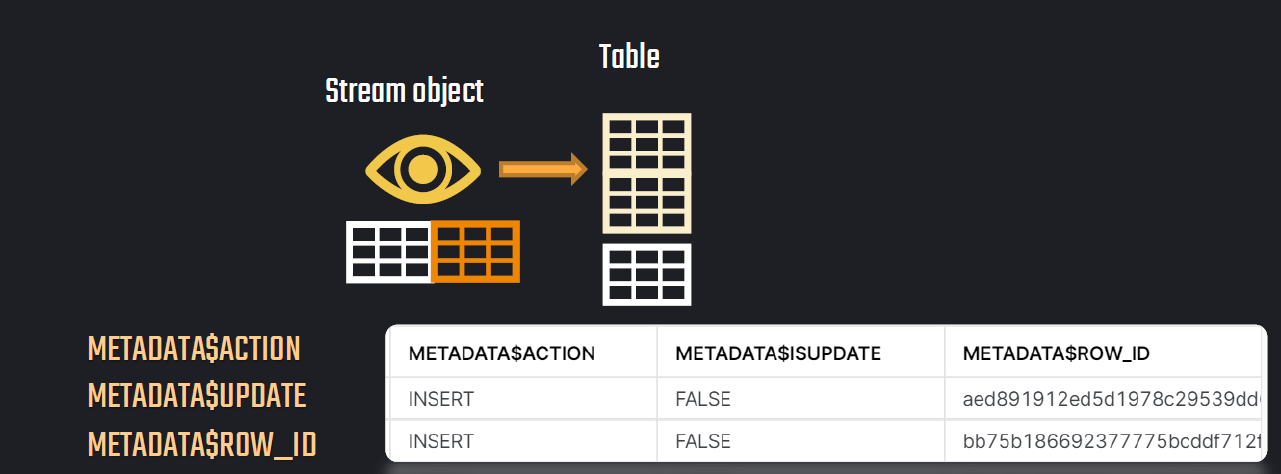


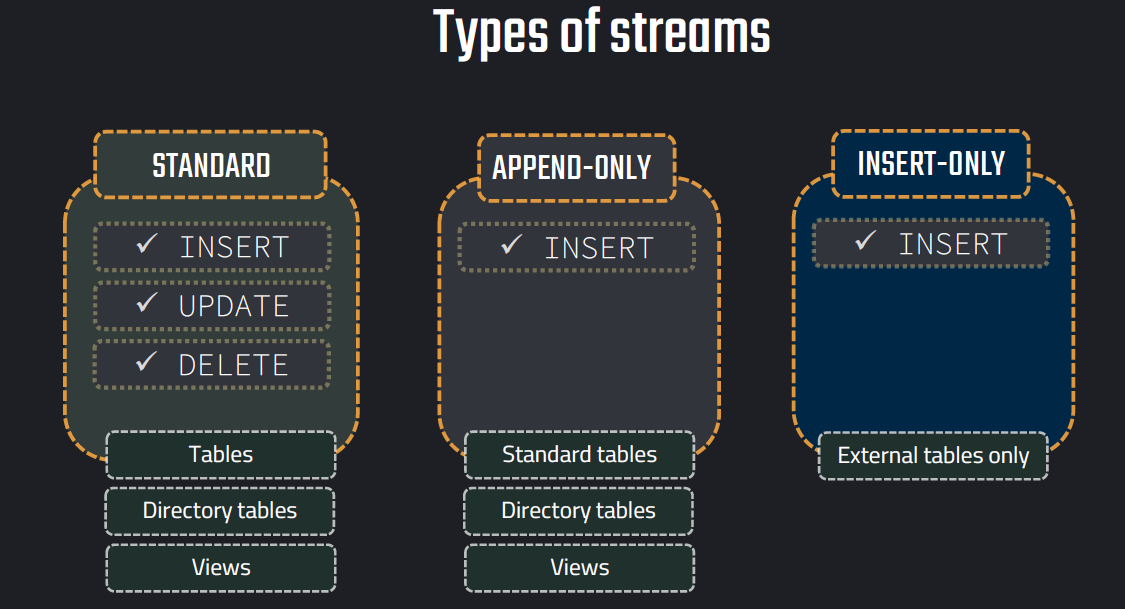


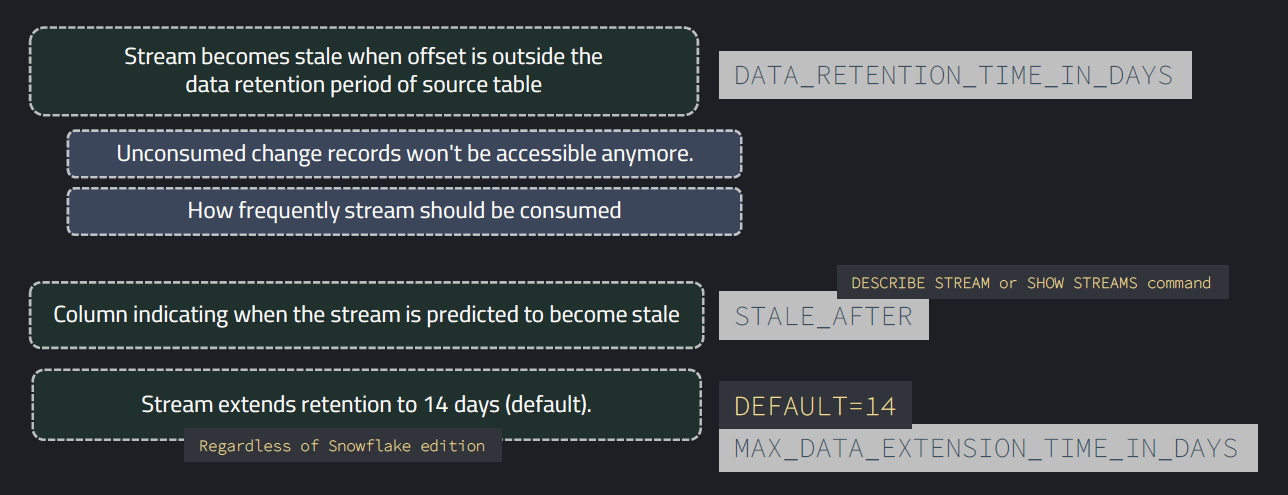


**Streams:**

A **Stream** is a powerful object that tracks **changes (inserts, updates, deletes)** on a table or view. It’s primarily used for **Change Data Capture (CDC)** scenarios — to identify **new, modified, or deleted rows** since the last time you queried the stream.







**Example:**

**-------------------- Stream example: INSERT ------------------------**

**CREATE OR REPLACE DATABASE STREAMS\_DB;**

**-- Create example table 1**

**create or replace table sales\_staging(**

**id varchar,**

**product varchar,**

**price varchar,**

**amount varchar,**

**store\_id varchar);**

**-- insert values**

**insert into sales\_staging**

**values**

**(1,'Banana',1.99,1,1),**

**(2,'Lemon',0.99,1,1),**

**(3,'Apple',1.79,1,2),**

**(4,'Orange Juice',1.89,1,2),**

**(5,'Cereals',5.98,2,1);**

**-- create second table**

**create or replace table store\_table(**

**store\_id number,**

**location varchar,**

**employees number);**

**INSERT INTO STORE\_TABLE VALUES(1,'Chicago',33);**

**INSERT INTO STORE\_TABLE VALUES(2,'London',12);**

**-- Create final input table**

**create or replace table sales\_final(**

**id int,**

**product varchar,**

**price number,**

**amount int,**

**store\_id int,**

**location varchar,**

**employees int);**

**-- Insert into final table**

**INSERT INTO sales\_final**

**SELECT**

**SA.id,**

**SA.product,**

**SA.price,**

**SA.amount,**

**ST.STORE\_ID,**

**ST.LOCATION,**

**ST.EMPLOYEES**

**FROM SALES\_STAGING SA**

**JOIN STORE\_TABLE ST ON ST.STORE\_ID=SA.STORE\_ID ;**

**SELECT \* FROM sales\_final;**

**-- Create a stream object**

**create or replace stream sales\_stream on table sales\_staging;**

**SHOW STREAMS;**

**DESC STREAM sales\_stream;**

**-- Get changes on data using stream (empty)**

**select \* from sales\_stream;**

**-- Query table**

**select \* from sales\_staging;**

**-- insert values**

**insert into sales\_staging**

**values**

**(6,'Mango',1.99,1,2),**

**(7,'Garlic',0.99,1,1);**

**-- Consume stream object**

**INSERT INTO sales\_final**

**SELECT**

**SA.id,**

**SA.product,**

**SA.price,**

**SA.amount,**

**ST.STORE\_ID,**

**ST.LOCATION,**

**ST.EMPLOYEES**

**FROM SALES\_STREAM SA**

**JOIN STORE\_TABLE ST ON ST.STORE\_ID=SA.STORE\_ID ;**

**-- Get changes on data using stream (INSERTS)**

**select \* from sales\_stream;**

**// Combine streams and tasks**

**CREATE OR REPLACE TASK sales\_stream\_task**

**WAREHOUSE = COMPUTE\_WH**

**SCHEDULE = '1 MINUTE'**

**WHEN SYSTEM$STREAM\_HAS\_DATA('SALES\_STREAM')**

**AS**

**INSERT INTO sales\_final**

**SELECT**

**SA.id,**

**SA.product,**

**SA.price,**

**SA.amount,**

**ST.STORE\_ID,**

**ST.LOCATION,**

**ST.EMPLOYEES**

**FROM SALES\_STREAM SA**

**JOIN STORE\_TABLE ST ON ST.STORE\_ID=SA.STORE\_ID ;**

**ALTER TASK sales\_stream\_task RESUME;**

**SHOW TASKS;**

**// Change data**

**INSERT INTO SALES\_STAGING VALUES (11,'Milk',1.99,1,2);**

**INSERT INTO SALES\_STAGING VALUES (12,'Chocolate',4.49,1,2);**

**INSERT INTO SALES\_STAGING VALUES (13,'Cheese',3.89,1,1);**

**// Verify results**

**SELECT \* FROM SALES\_STAGING;**

**SELECT \* FROM SALES\_STREAM;**

**SELECT \* FROM SALES\_FINAL;**