



INTRODUCTION

Section 0:
About the exam & course setup



01

02

03

04

05



01

02

03

04

05

06



About the **SNOWPRO CORE CERTIFICATION**

The Definite Preparation Course



01

02

03

04

05

06

01

02

03

04

05

06



Why getting certified?

- ✓ Impactful way to advance career
- ✓ Positioning as an expert
- ✓ Future proof + great job opportunities.

What is covered?

- ✓ SnowPro Core Certification
- ✓ <https://www.snowflake.com/certifications/>
- ✓ Topics are always kept up-to-date.

Demos

- ✓ Not needed for the exam.
- ✓ Help with memorizing.
- ✓ Give you practical foundation.

Goal

- ✓ Clear exam with ease.
- ✓ Knowledge for working with Snowflake.

Passing Score

- ✓ 750 / 1000
- ✓ Goal: Achieve a score of 900+.





How to Master the exam



01

02

03

04

05

06



01

02

03

04

05

06



Exam Topics

DOMAIN	WEIGHT	
1.0 Snowflake Data Cloud Features & Architecture	25%	A horizontal bar consisting of 10 segments, with the first 2.5 segments filled white and the remaining 7.5 segments unfilled.
2.0 Account Access and Security	20%	A horizontal bar consisting of 10 segments, with the first 2 segments filled white and the remaining 8 segments unfilled.
3.0 Performance Concepts	15%	A horizontal bar consisting of 10 segments, with the first 1.5 segments filled white and the remaining 8.5 segments unfilled.
4.0 Data Loading and Unloading	10%	A horizontal bar consisting of 10 segments, with the first 1 segment filled white and the remaining 9 segments unfilled.
5.0 Data Transformations	20%	A horizontal bar consisting of 10 segments, with the first 2 segments filled white and the remaining 8 segments unfilled.
6.0 Data Protection and Data Sharing	10%	A horizontal bar consisting of 10 segments, with the first 1 segment filled white and the remaining 9 segments unfilled.





Master the exam

Free Trial Account

Exam Overview

Exam Duration

Exam Questions

Not needed for the exam.
Help with memorizing.
Give you practical knowledge.

<https://www.snowflake.com/certifications/>

Time: 115min

100 questions Multiple Select, Multiple Choice

What are the features about? How do they work?

What is not a system role in Snowflake?

- SECURITYADMIN
- NETWORKADMIN
- ORGADMIN
- ACCOUNTADMIN

What is the mechanism to eliminate micro-partitions during query runtime?

- Pruning
- Caching
- Clustering
- Flattening



01

02

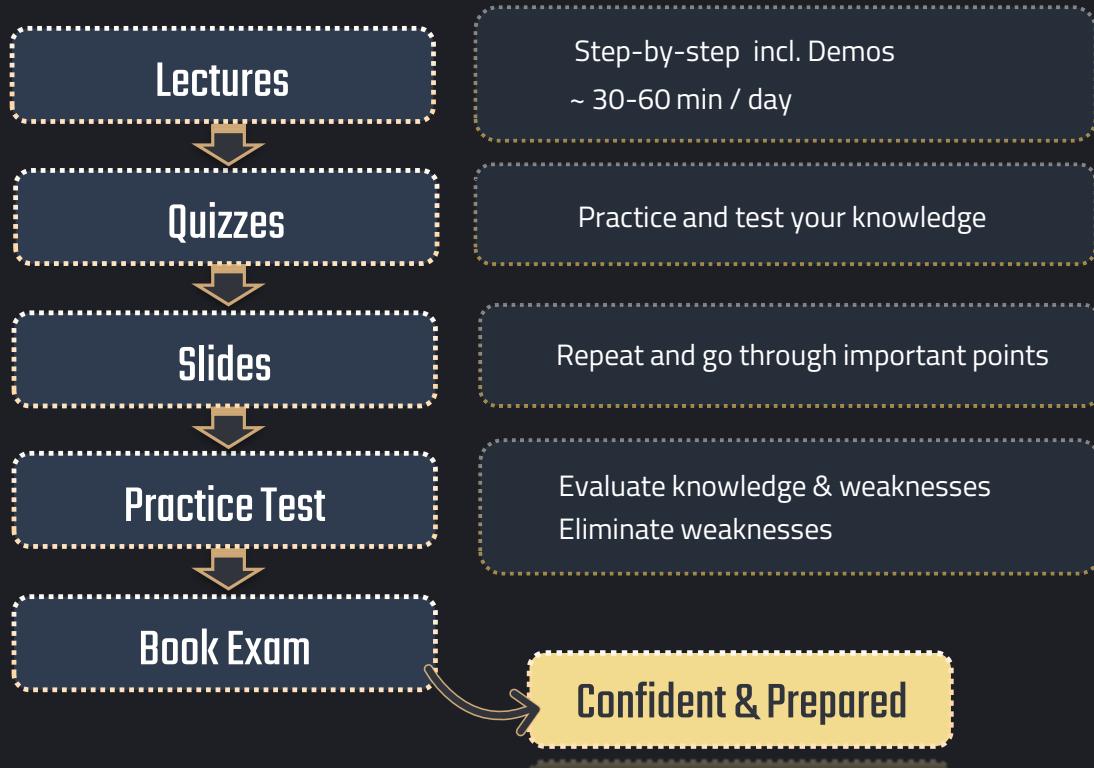
03

04

05

06

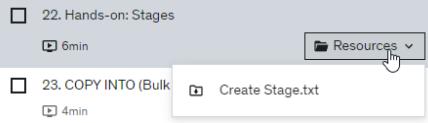
Recipe to clear the exam



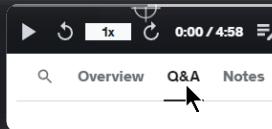


Final Tips

Resources



Q&A Section



Reviews



Connect & Congratulate





Snowflake Architecture

Domain 1.0:
Data Cloud Features & Architecture



01

02

03

04

05

06





What is Snowflake?



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





What is Snowflake?





73273

1760 0009-14563.7

1250 003-77156.8

003-1049559



Self-managed service



1

No Hardware

No need to select, install, configure, or manage.



2

No Software

No software needs to be installed, configured, or managed.



3

No Maintenance

Transparent Weekly releases.
No downtime.
Early access for Enterprise Edition accounts on request.



73273

1760 0009-14563.7

1250 003-77156.8

003-1040559



Cloud

Completely cloud-native.



Designed for Cloud

From scratch built for the cloud.



Runs in the Cloud

All components run completely in the cloud.
Cannot be installed on-premise.



Cloud optimized

Storage and compute scale independently and elastically.



73273

1760 0009-14563.7

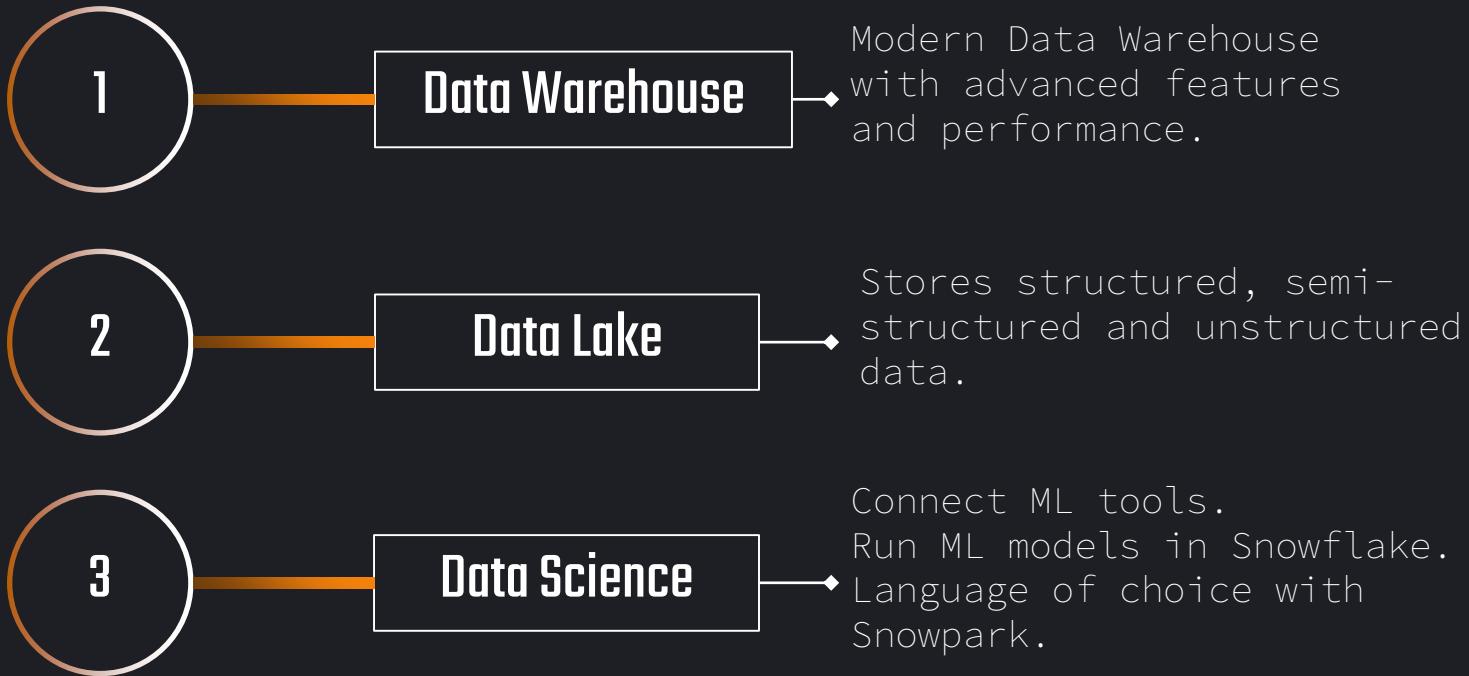
1250 003-77156.8

003-1049559



Data Platform

One single platform around data.





What is Snowflake?



T
3
✓

Self-managed

No Software,
No Hardware,
No Maintenance.

Cloud

Designed for Cloud,
Runs in the Cloud,
Optimized for Cloud.

Data Platform

Data Warehousing,
Data Engineering,
Data Applications.





Multi-cluster shared-disk



003-1040559

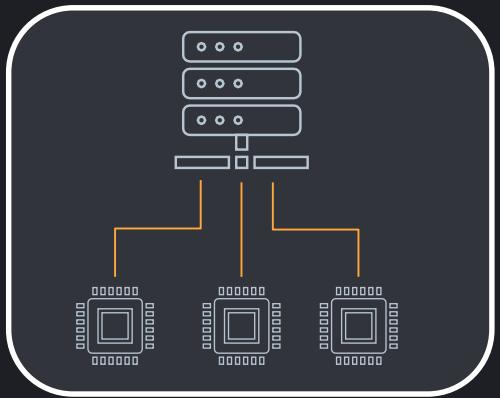
1250 003-77156.8

1760 0009-14563.7 73273



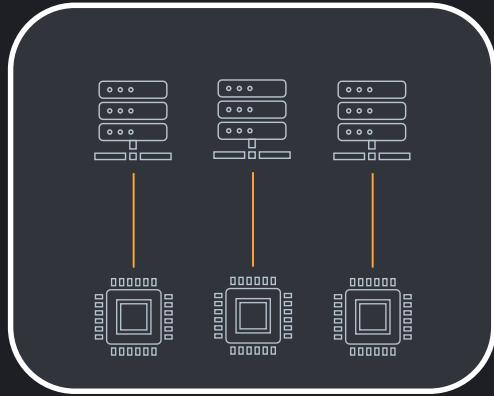


Traditional architectures



shared-disk

Central data storage Accessible from all compute nodes



shared-nothing

Each node is independent

Processor memory disk

01

02

03

04

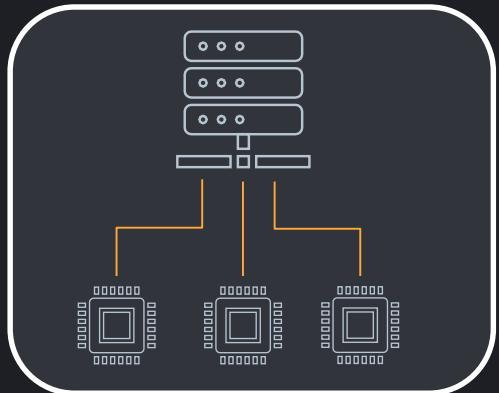
05

06





Traditional architectures



shared-disk

Limited scalability

Network bottleneck

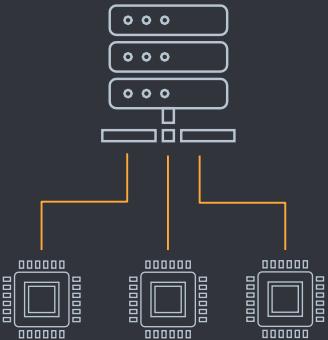
Single point of failure

PROs

Simplicity

Data management

CONs



shared-nothing

Management more difficult

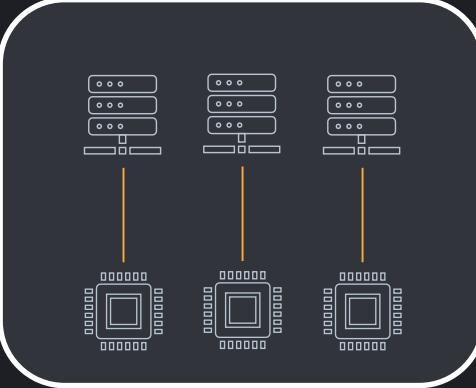
PROs

Scalability

High availability

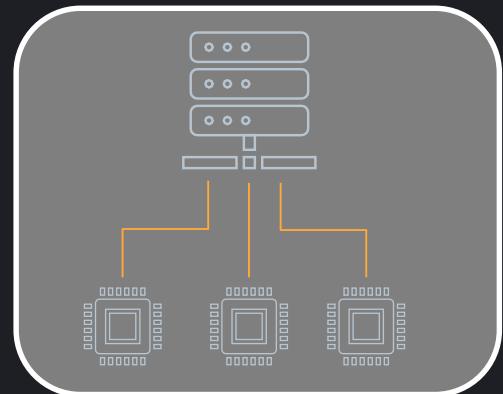
CONs

Expensive



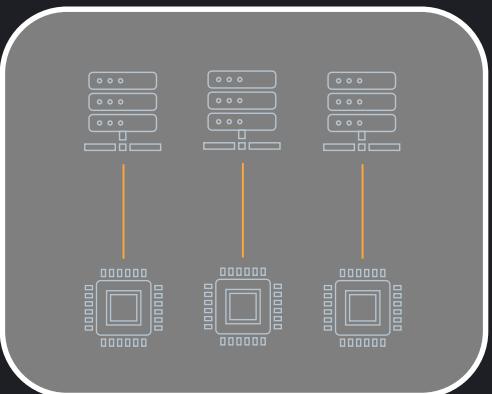


Traditional architectures



shared-disk

1



shared-nothing

2

Multi cluster shared-data

1

Central data repository

2

Massive parallel
processing compute
clusters

- Each node stores a portion of the data locally





Traditional architectures

PROs

Data management simplicity

performance
scale-out benefits

shared-disk

shared-nothing



Central data repository

Massive parallel
processing compute
clusters

—
each node stores a
portion of the data
locally

01

02

03

04

05

06

The background of the slide features a dark gray gradient with a subtle, glowing blue wavy pattern composed of many thin lines.

Three distinct layers



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Three distinct layers

CLOUD SERVICES



QUERY PROCESSING
(COMPUTE)



DATABASE STORAGE

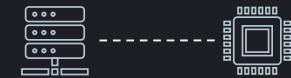




Three distinct layers

- Compressed Columnar Storage -

Decoupled Compute & Storage



Blob Storage (AWS, Azure, GCP)



Snowflake manages all aspects about storage



Optimized for OLAP / analytical purposes



DATABASE STORAGE





Three distinct layers

**QUERY PROCESSING
(COMPUTE)**

Virtual
Warehouse Virtual
Warehouse Virtual
Warehouse



DATABASE STORAGE





Three distinct layers

- "Muscle of the system" -

Queries are processed using Virtual warehouses

Warehouse = MPP compute cluster
(multiple compute nodes)



Provides resources: CPU, memory, and temporary storage

QUERY PROCESSING (COMPUTE)

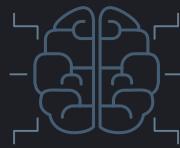
Virtual Warehouse Virtual Warehouse Virtual Warehouse





Three distinct layers

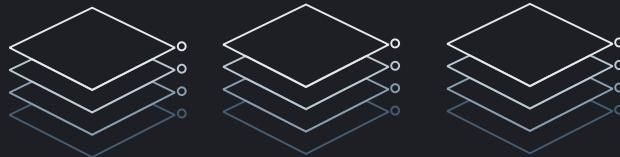
CLOUD SERVICES



QUERY PROCESSING
(COMPUTE)



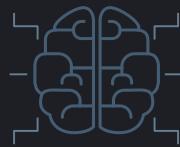
DATABASE STORAGE





Three distinct layers

CLOUD SERVICES



- "Brain of the system" -

- ✓ Authentication
- ✓ Access control
- ✓ Metadata management
- ✓ Query parsing and optimization
- ✓ Infrastructure management

Collection of services to coordinate & manage the components

Also run on compute instances of cloud provider





Three distinct layers

CLOUD SERVICES



QUERY PROCESSING
(COMPUTE)



DATABASE STORAGE





Snowflake Editions



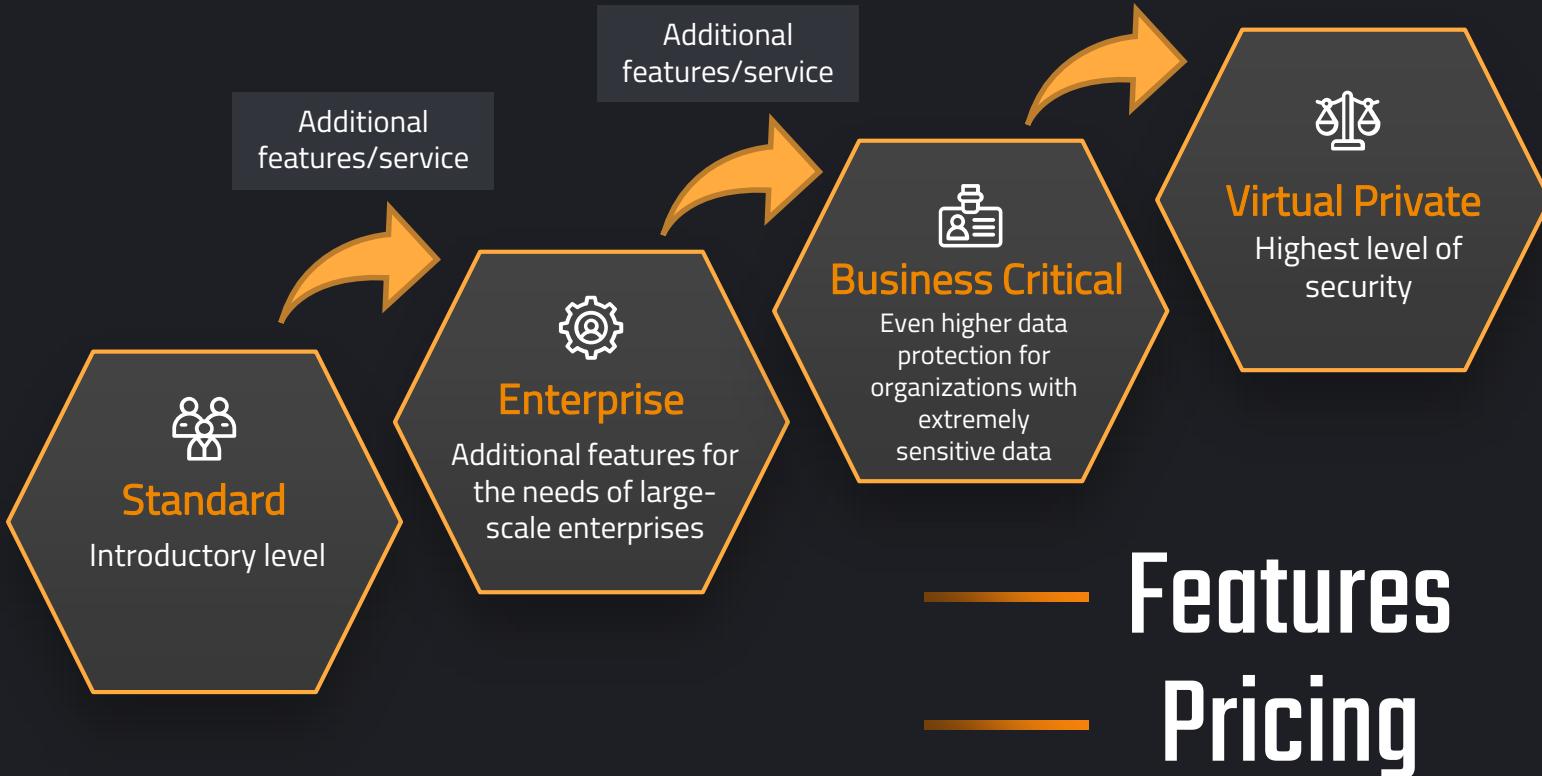
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273







Snowflake Editions



Standard

- ✓ Complete DWH
- ✓ Automatic data encryption
- ✓ Broad support for standard and special data types
- ✓ Time travel up to 1 day
- ✓ Disaster recovery for 7 days beyond time travel
- ✓ Network policies
- ✓ Secure data share
- ✓ Federated authentication & SSO
- ✓ Premier support 24/7



Enterprise

- ✓ All Standard features
- ✓ Multi-cluster warehouse
- ✓ Time travel up to 90 days
- ✓ Materialized views
- ✓ Search Optimization
- ✓ Column-level security
- ✓ 24-hour early access to weekly new releases



Business Critical

- ✓ All Enterprise features
- ✓ Additional security features such as customer-managed encryption
- ✓ Support for data specific regulation
- ✓ Database failover/failback (disaster recovery)



Virtual Private

- ✓ All Business Critical features
- ✓ Dedicated virtual servers and completely separate Snowflake environment
- ✓ Dedicated metadata store ⇒ Isolated from all other Snowflake accounts





Compute Cost



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing

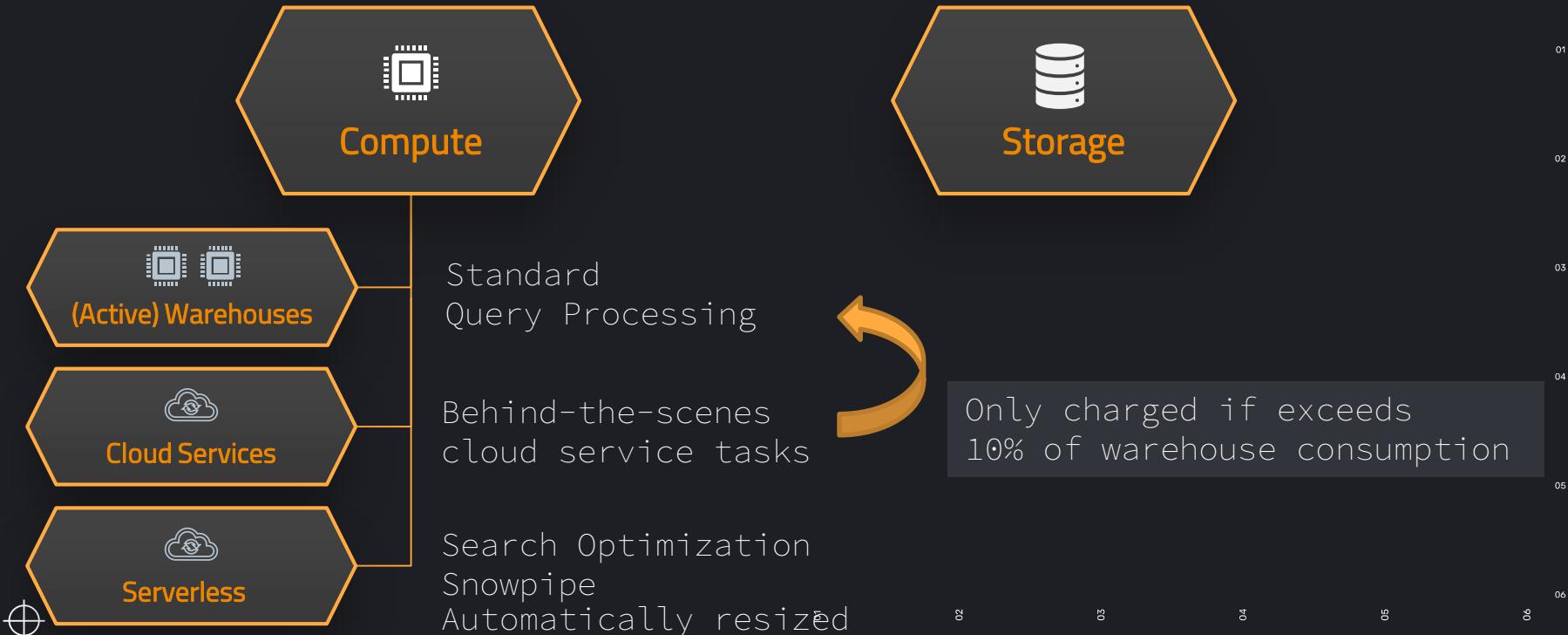


- ✓ Compute and Storage costs decoupled
- ✓ Pay only what you need
- ✓ Scalable and at affordable cloud price
- ✓ Pricing depending on the region/cloud provider



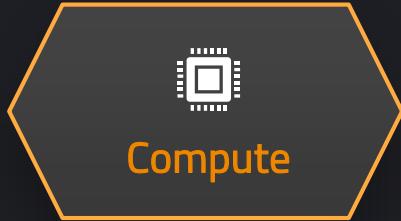


Snowflake Pricing





Snowflake Pricing



- ✓ Charged for active warehouses per hour
- ✓ Billed by second (minimum of 1min)
- ✓ Depending on the size of the warehouse

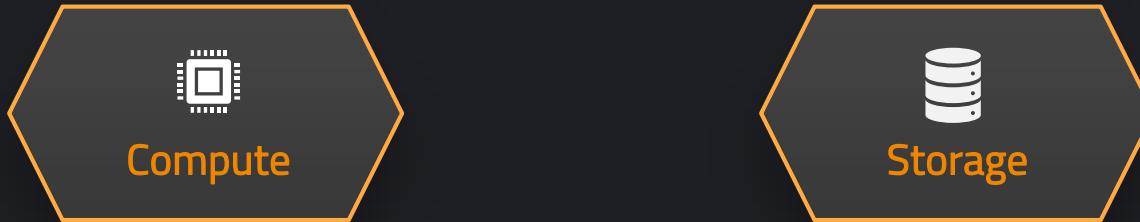
Time / active warehouses / size

- ✓ Charged in Snowflake credits





Snowflake Pricing



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Virtual Warehouse Sizes

XS **1**

S **2**

M **4**

L **8**

XL **16**

4XL **128**





Snowflake Pricing



Standard



Enterprise



Business Critical



Virtual Private

✓ \$2 / Credit

✓ \$3 / Credit

✓ \$4 / Credit

✓ Contact Snowflake

Region: US East (Ohio)

Platform: AWS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing



Standard



Enterprise



Business Critical



Virtual Private

✓ \$2.70 / Credit

✓ \$4 / Credit

✓ \$5.40 / Credit

✓ Contact Snowflake

Region: EU (Frankfurt)

Platform: AWS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Storage & Data Transfer



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Snowflake Pricing



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing

\$40

per TB / per month

Region: US East (Ohio)

Platform: AWS



Storage

- ✓ Monthly storage fees
- ✓ Based on average storage used per month
- ✓ Cloud Providers
- ✓ Cost calculated after compression





Snowflake Pricing



Storage



On Demand
Storage



Capacity
Storage

- ✓ Pay only for what you use
- ✓ \$40/TB
- ✓ Pay only for defined capacity upfront
- ✓ \$23/TB

Region: US East (Northern Virginia)

Platform: AWS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing



Storage



On Demand
Storage

- ✓ Pay only for what you use
- ✓ \$45/TB



Capacity
Storage

- ✓ Pay only for defined capacity upfront
- ✓ \$24.50/TB

Region: EU (Frankfurt)

Platform: AWS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing



✓ We think we need 1 TB of storage

- ❖ Scenario 1: 100GB of storage used
 $0.1 \text{ TB} \times \$40 = \4
- ❖ Scenario 2: 800GB of storage used
 $0.8 \text{ TB} \times \$40 = \32
- ❖ Scenario 1: 100GB of storage used
 $1 \text{ TB} \times \$23 = \23
- ❖ Scenario 2: 800GB of storage used
 $1 \text{ TB} \times \$23 = \23

Region: US East (Northern Virginia)

Platform: AWS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowflake Pricing



- ✓ Start with On Demand
- ✓ Once you are sure about your usage use Capacity storage





Snowflake Pricing



- ✓ Data ingress FREE
- ✓ Data egress CHARGED
- ✓ Cloud Storage Provider





Snowflake Pricing



- ✓ **Data ingress FREE**
- ✓ **Data egress CHARGED**
- ✓ **Cloud Storage Provider**

Transfer/replicate data to different account
(in different region and/or cloud provider)





Storage Monitoring



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Individual Table Storage

`SHOW TABLES;`

Statistics for table storage and properties

`TABLE_STORAGE_METRICS`
view in
`INFORMATION_SCHEMA`

`TABLE_STORAGE_METRICS`
view in
`ACCOUNT_USAGE`

Most detailed:
Active (ACTIVE_BYTES column)
Time Travel (TIME_TRAVEL_BYTES column)
Fail-safe (FAILSAFE_BYTES column)



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Individual Table Storage

SHOW TABLES;

SHOW TABLES;

TABLE_STORAGE_METRICS
view in
INFORMATION_SCHEMA

SELECT * FROM DB_NAME.INFORMATION_SCHEMA.TABLE_STORAGE_METRICS;

TABLE_STORAGE_METRICS
view in
ACCOUNT_USAGE

SELECT * FROM SNOWFLAKE.ACOUNT_USAGE.TABLE_STORAGE_METRICS;

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Resource Monitors



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Resource Monitors

Control and monitor credit usage of warehouses and account

Standard Edition

Credit Quota

Number of credits allowed per cycle

Set Credit Limits

Schedule

Start Monitoring **Immediately**
End Monitoring **Never**
Resets **Monthly**

Customize

In defined cycle



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Resource Monitors

Control and monitor usage of warehouses and account credits

Standard Edition

Virtual warehouse

Account

Multiple virtual warehouses

Monitor Type

Select Monitor Type



Schedule

Account

Warehouse

Warehouse

COMPUTE_WH, FIRST_WH

Schedule

COMPUTE_WH

FIRST_WH

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Resource Monitors

Control and monitor usage of warehouses and account credits

Standard Edition

Actions

Specify an action to perform when the quota is reached.

Add

Suspend immediately and notify

%

Suspend immediately and notify when this % of credit is used. [?](#)

Created by

ACCOUNTADMIN

Suspend and notify

%

Suspend and notify when this % of credit is used. [?](#)

Notify

%

Notify when this % of credit is used. [?](#)

MONITOR, MODIFY

On resource monitor



003-1040559

1250 003-77156.8

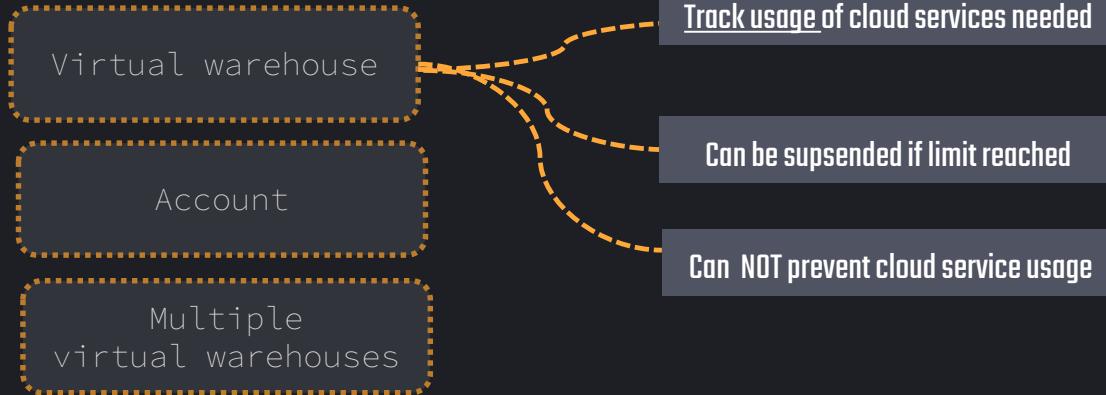
1760 0009-14563.7 73273



Resource Monitors

Control and monitor usage of warehouses and account credits

Standard Edition





Warehouses & Multi-Clustering



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Warehouses

What is a virtual warehouse?



Provides compute
resources to
execute queries
and operations

Type

Size

Multi-cluster





Warehouses



Standard

Most suitable in
most use cases



Snowpark-optimized

Recommended for
memory-intensive
workloads such as
ML training



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Virtual Warehouse Sizes

XS **1**

S **2**

M **4**

L **8**

XL **16**

4XL **128**





Virtual Warehouse Sizes

(Snowpark-optimized)

M



6

L



12

XL



24

6XL

768



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Multi-Clustering



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Multi-Clustering

... More queries ...



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Multi-Clustering

... More queries ...

> Auto-Scaling



Great for more concurrent users!

Not ideal for more complex workload!





Mode



Maximized

Min # clusters
=
Max # clusters

Static workload



Auto-scale

Min # clusters
≠
Max # clusters

Dynamic workload



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Multi-Clustering



Auto-Scaling: When to start an additional cluster?



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Scaling policy



Standard

Favors starting additional warehouses



Economy

Favors conserving credits rather than starting additional warehouses





Scaling policy

Policy	Description	Cluster Starts...	Cluster Shuts Down...
Standard (default)	Prevents/minimizes queuing by <u>favoring starting additional clusters</u> over conserving credits.	<u>Immediately</u> when either a query is queued or the system detects that there are more queries than can be executed by the currently available clusters.	<u>After 2 to 3 consecutive successful checks</u> (performed at 1 minute intervals), which determine whether the load on the least-loaded cluster could be redistributed to the other clusters
Economy	Conserves credits by favoring keeping running clusters fully-loaded rather than starting additional clusters. <u>Result:</u> May result in queries being queued and taking longer to complete.	Only if the system estimates there's enough query load to keep the cluster <u>busy for at least 6 minutes</u> .	<u>After 5 to 6 consecutive successful checks</u> ...





Snowflake Objects

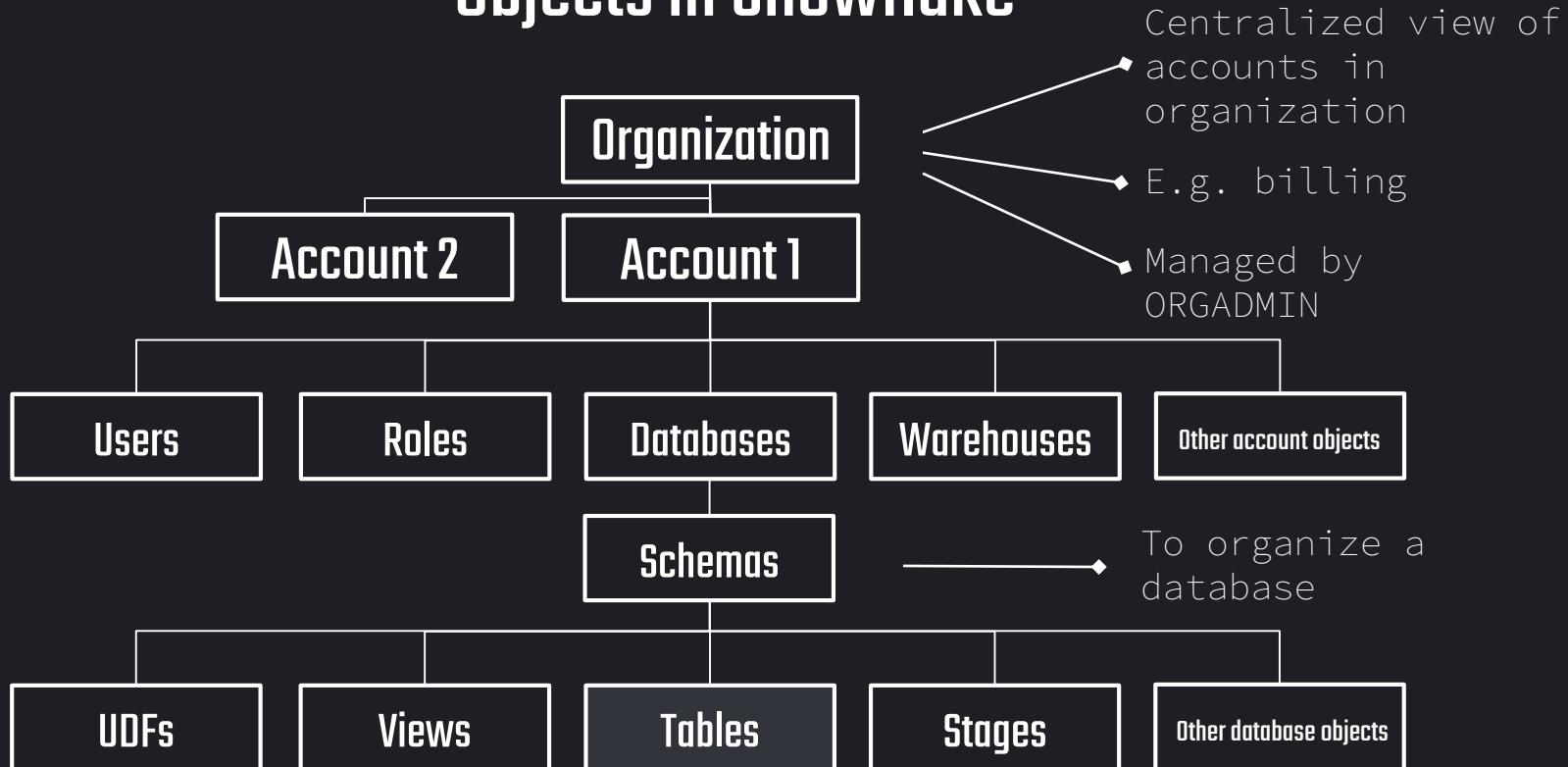


003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Objects in Snowflake





SnowSQL



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



SnowSQL

1

What is SnowSQL?

2

Download SnowSQL

3

Install SnowSQL

4

Connect to your Snowflake Account

5

Run a query in SnowSQL on your account



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



SnowSQL

Connect to Snowflake through the command line

Execute queries, load, and unload data

Perform all DDL & DML operations

Windows

Linux

MacOS





Data Loading & Unloading

Domain 4.0:
Data Loading and Unloading



01

02

03

04

05

06



01

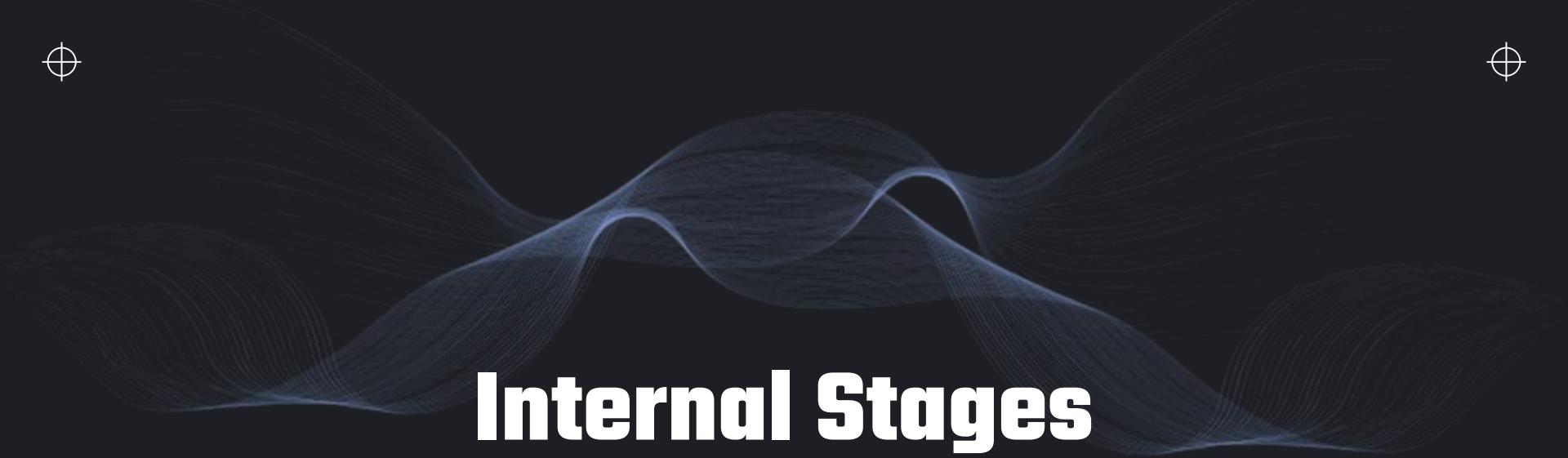
02

03

04

05

06



Internal Stages

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Stages



Stage



Database



Stages in Snowflake are locations used to store data.

- ✓ Location where data is loaded FROM





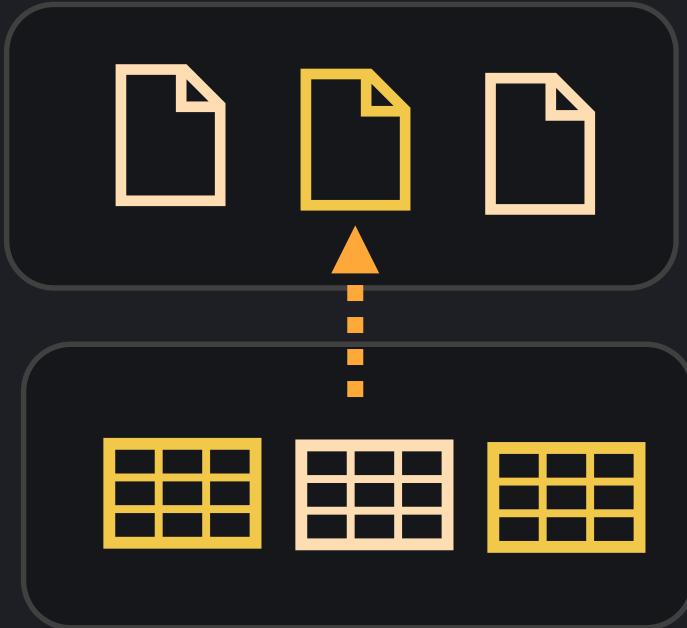
Stages



Stage



Database



Stages in Snowflake are locations used to store data.

- ✓ Location where data is loaded TO





Stages



Stage



Database

Stages in Snowflake are locations used to store data.

- ✓ Location where data is loaded FROM/TO
- ✓ Not to be confused with datawarehouse stages





Stages



Internal Stage Snowflake managed

- Cloud provider storage
- Upload file before load
- User stages
- Table stages
- Internal Named Stages

External Stage External cloud provider

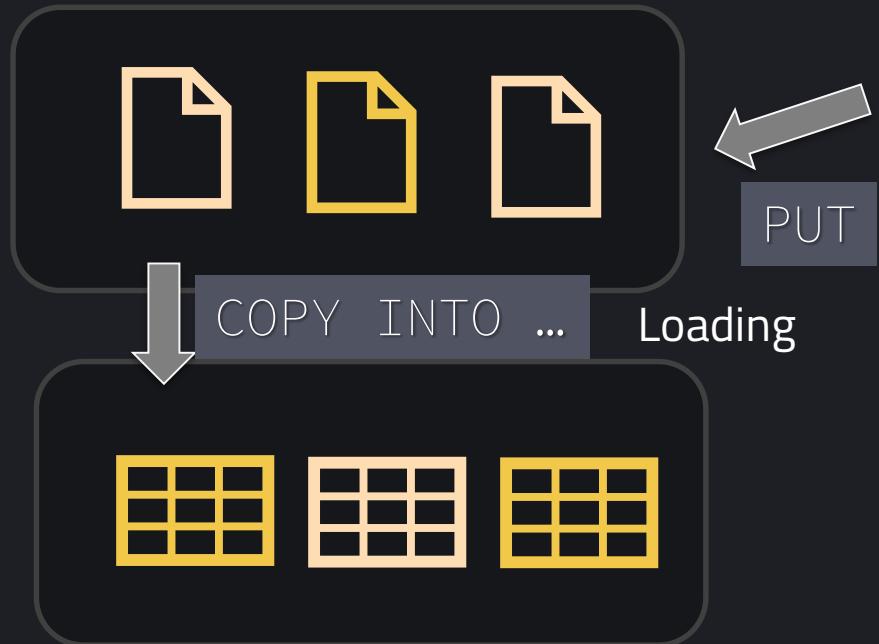
- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)





Stages

Internal
Stage



Tables

Uploading

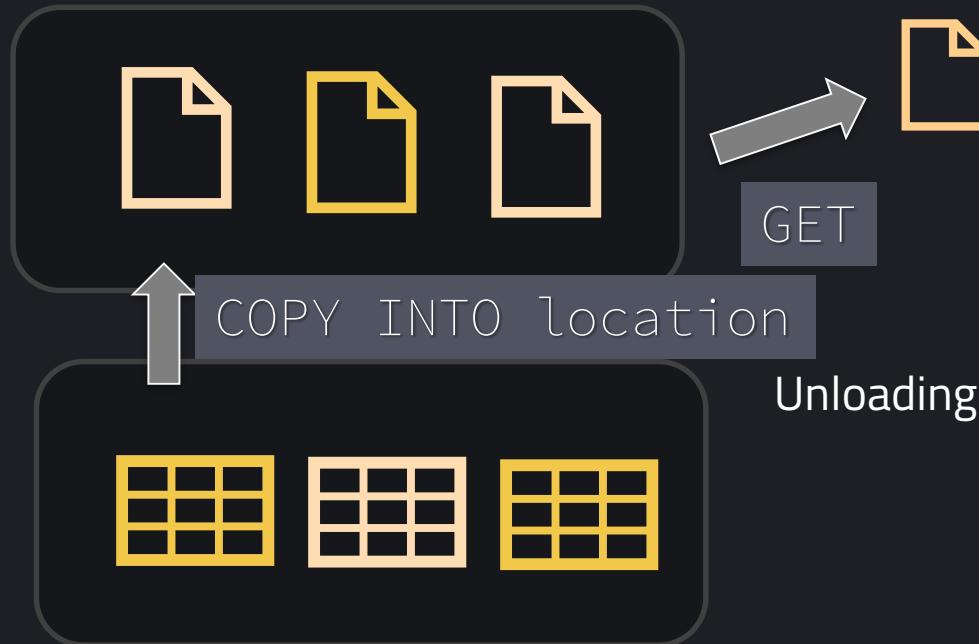
- Data will be compressed (.gz file ending)
- Automatically encrypted (128-bit or 256-bit keys)





Stages

Internal
Stage



Tables

Downloading

Unloading



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Stages

Internal Stage



User Stages

Table Stages

Internal Named Stages





Internal stages

User Stages

- Tied to a user
- Cannot be accessed by other users
- Every user has default stage
- Cannot be altered or dropped
- Put files to that stage before loading
- Explicitly remove files again
- Loading to multiple tables
- Referred to with '@~'





Internal stages

Table Stages

- Automatically created with a table
- Can only be accessed by one table
- Cannot be altered or dropped
- Load to one table
- Referred to with '@%TABLE_NAME'





Internal stages

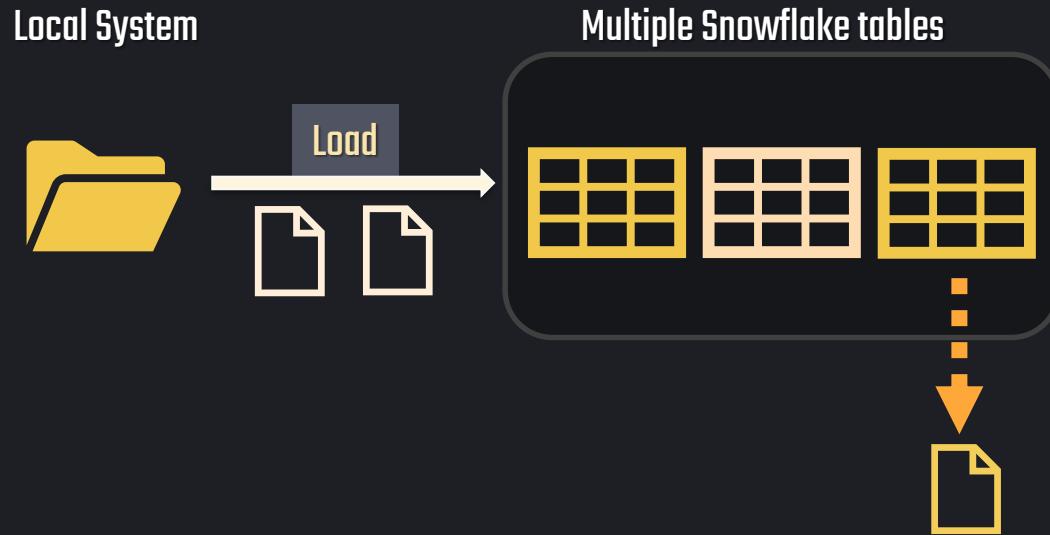
Named Stages

- CREATE STAGE ...
- Snowflake database object
- Everyone with privileges can access it
- Most flexible
- Referred to with '@STAGE_NAME'





Use cases



003-1040559

1250 003-77156.8

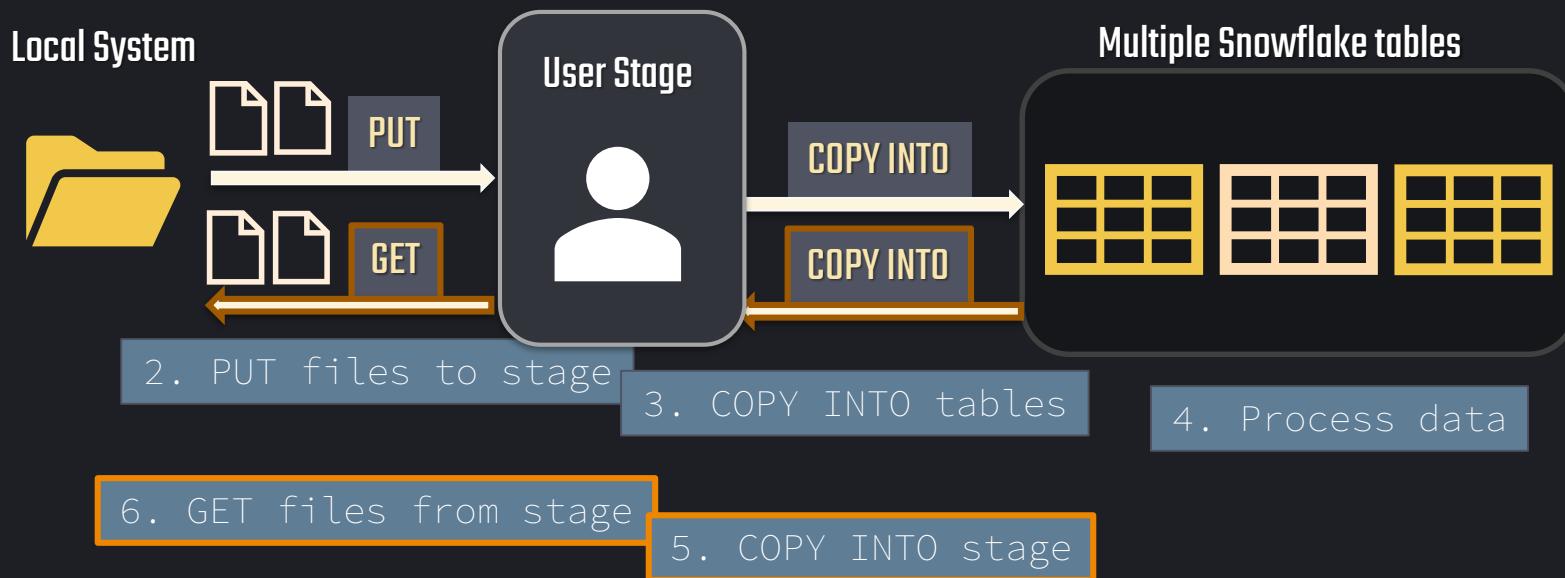
1760 0009-14563.7 73273





Use cases

1. Connect to SnowSQL





External Stages

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Stages



Internal Stage Snowflake managed

- Cloud provider storage
- Upload file before load
- User stages
- Table stages
- Internal Named Stages

External Stage External cloud provider

- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)





Stages

External Stage



External cloud provider

- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)

- CREATE STAGE ...
- Snowflake database object
- Everyone with privileges can access it
- Referred to with '@STAGE_NAME'
- References external storage location

```
CREATE STAGE <stage_name>
URL = 's3://bucket/path/'
```





Stages

External Stage



External cloud provider

- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)

- CREATE STAGE ...
- Snowflake database object
- Everyone with privileges can access it
- Referred to with '@STAGE_NAME'
- References external storage location

```
CREATE STAGE <stage_name>
URL = 'azure://account.blob.core.windows.net/container/path/'
```





Stages

External Stage



External cloud provider

- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)

- CREATE STAGE ...
- Snowflake database object
- Everyone with privileges can access it
- Referred to with '@STAGE_NAME'
- References external storage location

```
CREATE STAGE <stage_name>
URL = 's3://bucket/path/'
CREDENTIALS = ...
FILE_FORMAT = ...
```





Stages

External Stage



External cloud provider

- AWS S3 (AWS)
- Google Cloud Storage (GCP)
- Azure Container (Azure)

- CREATE STAGE ...
- Snowflake database object
- Everyone with privileges can access it
- Referred to with '@STAGE_NAME'
- References external storage location

```
CREATE STAGE <stage_name>
URL = 's3://bucket/path/'
STORAGE_INTEGRATION = ...
FILE_FORMAT = ...
```





LIST

```
LIST @STAGE_NAME;
```

External Stage / Internal Named Stage

```
LIST @~;
```

User Stage

List all files
and additional properties

```
LIST @%TABLE_STAGE_NAME;
```

Table Stage





Referencing stages

Copy FROM stage

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME;
```

Query from stage

```
SELECT * FROM @STAGE_NAME;
```

Copy TO stage

```
COPY INTO @STAGE_NAME  
FROM TABLE_NAME;
```

Table Stage

```
SELECT  
$1,  
$2,  
$3  
FROM @STAGE_NAME;
```





COPY INTO



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





COPY INTO

Copy FROM stage

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME;
```

- Loads data into a table
Files must be already staged in:
 - Internal or External Stage

Copy TO stage

```
COPY INTO @STAGE_NAME  
FROM TABLE_NAME;
```

- Unloads data into a file





COPY INTO

Copy FROM stage

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME;
```

File formats:

- CSV (*Default*)
- JSON
- AVRO
- ORC
- PARQUET
- XML

- Loads data into a table
 ⇒ Bulk Loading
- Files must be already staged in:
 - Internal Named Stage
 - External Stage
- Warehouses are needed
- Data Transfer costs may apply





COPY INTO

Load specific files

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME  
FILES = file_name1, ...;
```

Load with pattern

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME  
PATTERN = .*sales.*;
```

Load with Copy Options

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME  
FILES = file_name1, ...  
CopyOptions;
```





COPY INTO

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME;  
FILE_FORMAT = ( FORMAT_NAME = 'file_format_name' |  
                TYPE = CSV | JSON | AVRO | ORC | PARQUET | XML)
```





File Format



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





FILE FORMAT

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME;
```

```
COPY INTO TABLE_NAME  
FROM @STAGE_NAME  
FILE_FORMAT = (TYPE = CSV ...);
```

DESC STAGE manage_db.external_stages.aws_stage

Objects Editor Results Chart

property	property	...	property_type	property_value	property_default
FILE_FORMAT	TYPE		String	CSV	CSV
FILE_FORMAT	RECORD_DELIMITER		String	\n	\n
FILE_FORMAT	FIELD_DELIMITER		String	,	,
FILE_FORMAT	FILE_EXTENSION		String		
FILE_FORMAT	SKIP_HEADER		Integer	0	0
FILE_FORMAT	DATE_FORMAT		String	AUTO	AUTO
FILE_FORMAT	TIME_FORMAT		String	AUTO	AUTO
FILE_FORMAT	TIMESTAMP_FORMAT		String	AUTO	AUTO





FILE FORMAT

```
CREATE FILE FORMAT <fileformatname>
    TYPE = CSV
    FIELD_DELIMITER = ','
    SKIP_HEADER = 1;
```

DESC STAGE manage_db.external_stages.aws_stage

The screenshot shows a table with the following data:

property	property	...	property_type	property_value	property_default
FILE_FORMAT	TYPE		String	CSV	CSV
FILE_FORMAT	RECORD_DELIMITER		String	\n	\n
FILE_FORMAT	FIELD_DELIMITER		String	,	,
FILE_FORMAT	FILE_EXTENSION		String		
FILE_FORMAT	SKIP_HEADER		Integer	0	0
FILE_FORMAT	DATE_FORMAT		String	AUTO	AUTO
FILE_FORMAT	TIME_FORMAT		String	AUTO	AUTO
FILE_FORMAT	TIMESTAMP_FORMAT		String	AUTO	AUTO

```
CREATE STAGE <stagename>
URL = '<location>'
FILE_FORMAT = (FORMAT_NAME = <fileformatname >);
```





FILE FORMAT

```
CREATE FILE FORMAT <fileformatname>
  TYPE = CSV
  FIELD_DELIMITER = ','
  SKIP_HEADER = 1;
```

```
CREATE STAGE <stagename>
  URL = '<location>'
  FILE_FORMAT = (FORMAT_NAME = <fileformatname >);
```

```
COPY INTO table_name
  FROM @<stagename>;
```

```
COPY INTO table_name
  FROM @<stagename>
  FILE_FORMAT = (TYPE = CSV ...);
```

```
COPY INTO table_name
  FROM @<stagename>
  FILE_FORMAT = (FORMAT_NAME = <fileformatname >);
```





Storage Integration



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Integration object

Storage Integration

- ✓ Stores a generated identity for external cloud storage

Create Snowflake object

- ✓ Contains ALLOWED_LOCATIONS

Grant permission in Cloud Provider

- ✓ Allow it as Enterprise Application

Assign Role in Cloud Provider

- ✓ Grant permissions

Use it in Stage

- ✓ Connect it to stage object





Snowpipe



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Loading Data

BULK LOADING

- ✓ Manual execution

CONTINUOUS LOADING

- ✓ Snowpipe



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



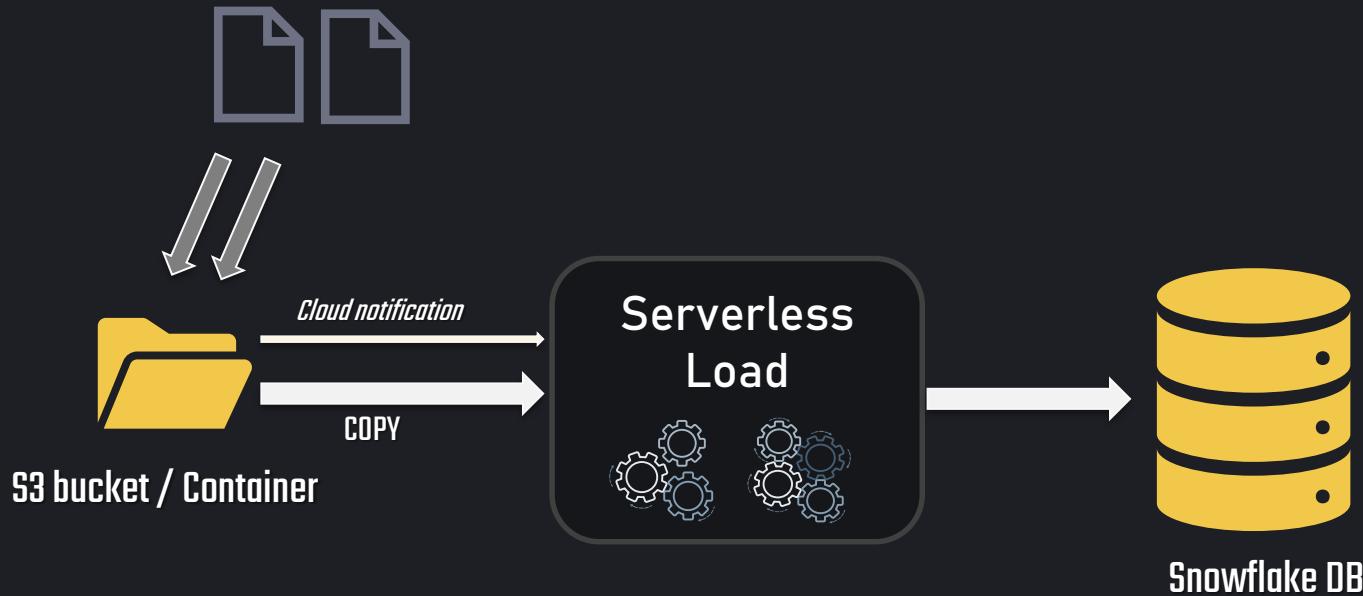
What is Snowpipe?

- ✓ Loads data immediately when a file appears in a blob storage
- ✓ According to defined COPY statement
- ✓ If data needs to be available immediately for analysis
- ✓ Snowpipe uses serverless features instead of warehouses
 - ⇒ No user-created warehouse is needed!





Snowpipe



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



01

02

03

04

05

06

01

02

03

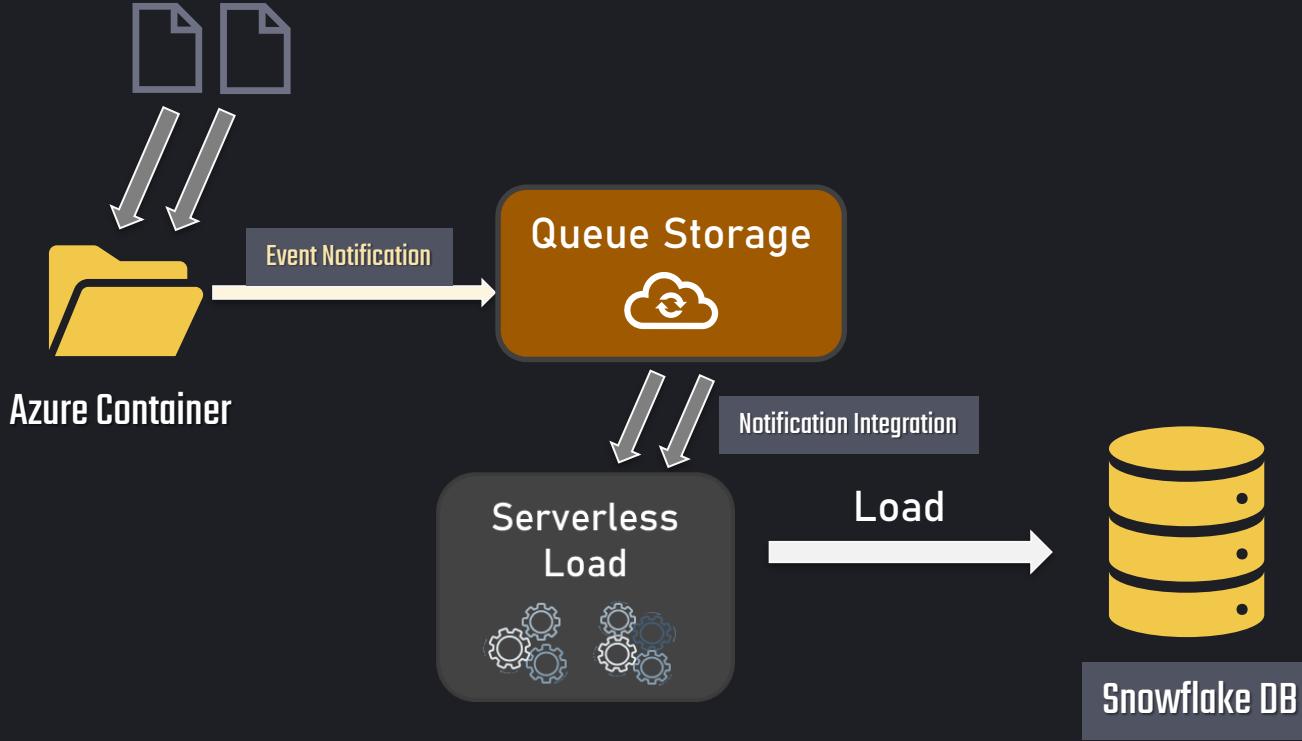
04

05

06



Snowpipe for Azure





01

02

03

04

05

06



01

02

03

04

05

06

Snowpipe for Azure

```
CREATE PIPE <name>
AUTO_INGEST = TRUE | FALSE
INTEGRATION = '<string>'
COMMENT = '<string_literal>'
AS <copy_statement>
```

- ✓ PIPE contains COPY statement





01

02

03

04

05

06



01

02

03

04

05

06

Snowpipe for Azure

```
CREATE PIPE <name>
AUTO_INGEST = TRUE | FALSE
INTEGRATION = '<string>'
COMMENT = '<string_literal>'
AS COPY INTO table_name
FROM @stage_name
```

- ✓ PIPE contains COPY statement





Setting up Snowpipe

Storage Integration

- ✓ Connection details to container
- ✓ Grant permissions

Create Stage

- ✓ Location to container

Queue + Notification

- ✓ To trigger snowpipe

Notification Integration

- ✓ Notification can be received by Snowflake
- ✓ Grant permissions

Create Pipe

- ✓ Create pipe as object with COPY COMMAND

Test COPY COMMAND

- ✓ To make sure it works





Snowpipe

Snowpipe Methods

Cloud messaging

- ✓ Uses event notifications
- ✓ External stages

REST API

- ✓ Calls REST API Endpoints
- ✓ Internal stages & External stages





Snowpipe

Serverless



No dedicated warehouse

Cost



Per-second/per-core granularity

Load Time



Typically within 1 minute

File Size



Ideally 100MB - 250MB (or more)



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowpipe

Load History



14 Days

Location



Store in Schema

Pause / Resume



ALTER PIPE
SET PIPE_EXECUTION_PAUSED = TRUE



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Copy Options



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Copy Options

```
COPY INTO <table_name>
FROM @externalStage
FILES = ('<file_name>', '<file_name2>')
FILE_FORMAT = <file_format_name>
copyOptions
```

How data is copied

Unloading vs. Loading



Copy Options

Stage properties

	parent_property	property	property_type	property_value	property_default
1	STAGE_FILE_FORMAT	FORMAT_NAME	String	fileformat_azure	
2	STAGE_COPY_OPTIONS	ON_ERROR	String	ABORT_STATEMENT	ABORT_STATEMENT
3	STAGE_COPY_OPTIONS	SIZE_LIMIT	Long		
4	STAGE_COPY_OPTIONS	PURGE	Boolean	false	false
5	STAGE_COPY_OPTIONS	RETURN_FAILED_ONLY	Boolean	false	false
6	STAGE_COPY_OPTIONS	ENFORCE_LENGTH	Boolean	true	true
7	STAGE_COPY_OPTIONS	TRUNCATECOLUMNS	Boolean	false	false
8	STAGE_COPY_OPTIONS	FORCE	Boolean	false	false
9	STAGE_LOCATION	URL	String	["azure://demosnowflake"]	



Copy Options

```
CREATE STAGE <stage_name>
URL = 's3://bucket/path/'
STORAGE_INTEGRATION = ...
FILE_FORMAT = ...
COPY_OPTIONS = ( copyOptions )
```





Copy Options

```
COPY INTO <table_name>
FROM @externalStage
FILES = ('<file_name>', '<file_name2>')
FILE_FORMAT = <file_format_name>
copyOptions
```

If specified in COPY command they will overwrite the stage copy options





Copy Options – ON ERROR

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
ON_ERROR = CONTINUE
```

Only with Data Loading

CONTINUE

Continue loading file if errors are found

DEFAULT
SNOWPIPE

SKIP_FILE

Skip file if errors are found

SKIP_FILE_<num>

e.g. SKIP_FILE_10: Skip file when # errors \geq 10

SKIP_FILE_<num>%

e.g. SKIP_FILE_10%: Skip file when # errors \geq 10%

DEFAULT
BULK LOAD

ABORT_STATEMENT

Aborts load if error is found





Copy Options – SIZE_LIMIT

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
SIZE_LIMIT = <num>
```

SIZE_LIMIT = <num>

DEFAULT
null

Specifies max. size of data loaded

Next file will be loaded until <num> (in bytes) is exceeded





Copy Options – SIZE_LIMIT

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
SIZE_LIMIT = 25000000
```





Copy Options – PURGE

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
PURGE= TRUE | FALSE
```

TRUE

Remove files (if possible) from stage after
successful load

DEFAULT

FALSE

Leave files in stage





Copy Options – MATCH_BY_COLUMN_NAME

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>' , '<file_name2>')
MATCH_BY_COLUMN_NAME = CASE_SENSITIVE | CASE_INSENSITIVE | NONE
```

Load semi-structured data columns by matching field names

CASE_SENSITIVE

Matches case-sensitive

CASE_INSENSITIVE

Matches case-insensitive

DEFAULT

NONE

Loads data in variant column or based on COPY statement





Copy Options – ENFORCE_LENGTH

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
ENFORCE_LENGTH = TRUE | FALSE
```

DEFAULT

TRUE

Produces error if loaded string length is exceeded

FALSE

Strings are automatically truncated





Copy Options – TRUNCATECOLUMNS

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
TRUNCATECOLUMNS = TRUE | FALSE
```

TRUE

Strings are automatically truncated

DEFAULT

FALSE

Produces error if loaded string length is exceeded





Copy Options – FORCE

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
FORCE = TRUE | FALSE
```

TRUE

Load files even if they have been loaded before

DEFAULT

FALSE

Don't load file when they have been loaded before





Copy Options – LOAD_UNCERTAIN_FILES

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
LOAD_UNCERTAIN_FILES = TRUE | FALSE
```

TRUE

Load file if load status is unknown

DEFAULT

FALSE

Don't load file if load status is unknown





Copy Options

CopyOption	Description	Values	Default
ON_ERROR	Specifies the error handling for the load operation	CONTINUE SKIP_FILE SKIP_FILE_num 'SKIP_FILE_num%' ABORT_STATEMENT	ABORT_STATEMENT
SIZE_LIMIT	Specifies the maximum size (in bytes) of data to be loaded	<num>	null (no limit)
PURGE	Remove files after successful load	TRUE FALSE	FALSE
RETURN_FAILED_ONLY	Return only files that have failed to load	TRUE FALSE	FALSE
MATCH_BY_COLUMN_NAME	Load semi-structured data into columns in matching the column names	CASE_SENSITIVE CASE_INSENSITIVE NONE	NONE
ENFORCE_LENGTH	Truncate text strings that exceed the target column length	TRUE FALSE	TRUE
TRUNCATECOLUMNS	Truncate text strings that exceed the target column length	TRUE FALSE	FALSE
FORCE	Load files even if loaded before	TRUE FALSE	FALSE
LOAD_UNCERTAIN_FILES	Load files even if load status unknown	TRUE FALSE	FALSE





VALIDATION_MODE



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



VALIDATION_MODE

```
COPY INTO <table_name>
FROM externalStage
FILES = ('<file_name>', '<file_name2>')
VALIDATION_MODE = RETURN_n_ROWS | RETURN_ERRORS
```

VALIDATE THE DATA INSTEAD OF LOADING

RETURN_n_ROWS

e.g. RETURN_5_ROWS: Validates n rows (returns error or rows)

RETURN_ERRORS

Returns all errors across all files





VALIDATE



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



VALIDATE

```
VALIDATE( <table_name> , JOB_ID => { '<query_id>' | '_last' } )
```

Validates the files loaded in a past execution of the COPY INTO <table> command and returns all the errors encountered during the load

Doesn't return anything if ON_ERROR = ABORT_STATEMENT

```
SELECT * FROM TABLE(VALIDATE( ORDERS , JOB_ID => '_last' ))
```



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Unloading



003-1040559

1250 003-77156.8

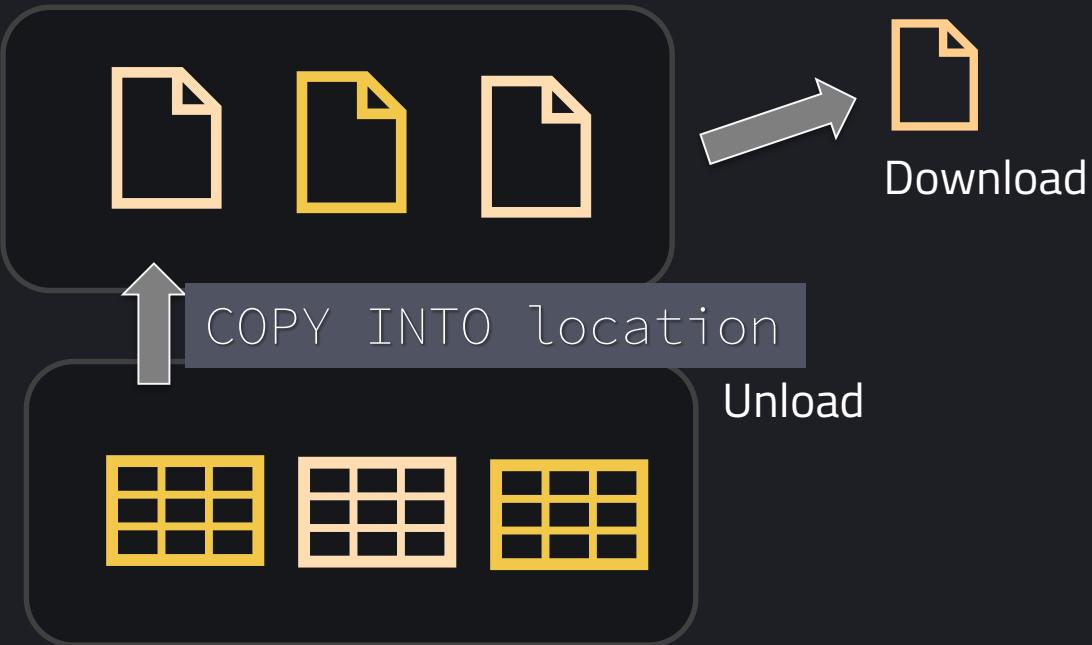
1760 0009-14563.7

73273



Unloading

Stage



Tables

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273





Unloading

Internal
Stage



003-1040559

1250 003-77156.8

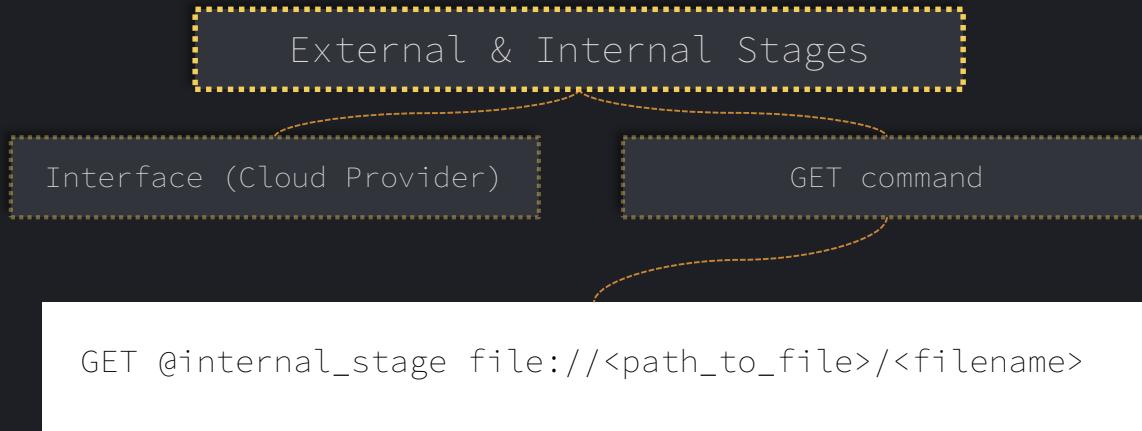
1760 0009-14563.7 73273





Unloading

```
COPY INTO @stage_name  
FROM table_name
```



File formats:

Structured

- CSV (Default), TSV, etc.

Semi-structured:

- JSON
- PARQUET
- XML





Unloading

```
COPY INTO @stage_name  
FROM table_name
```



SINGLE = FALSE

DEFAULT

Split into multiple files

SINGLE = TRUE

Everything in one file





Unloading

```
COPY INTO @stage_name  
FROM table_name
```



MAX_FILE_SIZE <num>

DEFAULT
=16777216

16MB

Can be increased to 5 GB





Unloading

```
COPY INTO @stage_name  
FROM (SELECT id, name, start_date  
      FROM table_name)
```



Unload using SELECT

SELECT statement can be used for the COPY statement

Transformations

Transformations can be used





Unloading

```
COPY INTO @stage_name  
FROM (SELECT id, name, start_date  
      FROM table_name)  
FILE_FORMAT=(TYPE=JSON)
```



FILE_FORMAT

File format can be set in the COPY command





Unloading

```
COPY INTO @stage_name  
FROM (SELECT id, name, start_date  
      FROM table_name)  
FILE_FORMAT=(TYPE=CSV)  
HEADER = TRUE
```



FILE_FORMAT

File format can be set in the COPY command

HEADER = TRUE

Will include a header in the output files





Unloading

```
COPY INTO @stage_name/myfile
FROM (SELECT id, name, start_date
      FROM table_name)
FILE_FORMAT=(TYPE=CSV)
HEADER = TRUE
```



Prefix

If not specified 'data_' is prefix

Suffix

Suffix is added to ensures each file name is unique

e.g. data_0_1_0.csv.gz

e.g. myfile_0_0_1. csv.gz



Data Transformations

Domain 5.0:
Data Transformations



03

04

05

06



01

02

03

04

05

06



Transformations & Functions





Data Transformations

Data can be transformed when loading data

Simplify ETL pipeline

Supported

Column reordering

Cast data types

Remove columns

Truncate
(TRUNCATECOLUMNS)

Subset of SQL functions

Not Supported

FLATTEN function

Aggregation functions

GROUP BY

Filter with WHERE

JOINS





Functions

Supports most standard SQL functions defined in SQL:1999 and parts of SQL:2003 extensions

Scalar functions

Returns one value per invocation (one value per row)

```
SELECT DAYNAME('2023-12-31')
```

DAYNAME('2023-12-31')

1	Sun
---	-----

```
SELECT DAYNAME("effective_date")
FROM LOAN_PAYMENT
```

DAYNAME("EFFECTIVE_DATE")

1	Thu
2	Thu
3	Thu
4	Thu
5	Fri



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

2 44
Unit



Functions

Supports most standard SQL functions defined in SQL:1999

Scalar functions

Returns one value per invocation (one value per row)

Aggregate functions

Mathematical calculations such as max & min across rows

```
SELECT MAX(amount)  
FROM orders
```

MAX(AMOUNT)

1 98





Functions

Supports most standard SQL functions defined in SQL:1999

Scalar functions

Returns one value per invocation (one value per row)

Aggregate functions

Mathematical calculations such as max & min across rows

Window functions

Aggregate functions that operate on a subset of rows

```
SELECT ORDER_ID, SUBCATEGORY,  
MAX(AMOUNT) OVER (PARTITION BY SUBCATEGORY)  
FROM ORDERS
```

	ORDER_ID	SUBCATEGORY	MAX(AMOUNT) OVER (P
1	B-30601	Stole	97
2	B-30601	Electronic Games	98
3	B-30602	Phones	977



003-1040559

1250 003-77156.8

1760 0009-14563.7 B-73273

Phones

811



Functions

Supports most standard SQL functions defined in SQL:1999

Scalar functions

Returns one value per invocation (one value per row)

Aggregate functions

Mathematical calculations such as max & min across rows

Window functions

Aggregate functions that operate on a subset of rows

Table functions

Return a set of rows for each input row – used to obtain information about Snowflake features

```
SELECT * FROM TABLE(VALIDATE( ORDERS , JOB_ID => '_last' ))
```

	ERROR	FILE
1	Numeric value 'one thousand' is not recognized	returnfailed/OrderDetails_error.csv
2	Numeric value 'two hundred twenty' is not recognized	returnfailed/OrderDetails_error.csv
3	Numeric value '7-' is not recognized	returnfailed/OrderDetails_error2 - Copy.c





Functions

Supports most standard SQL functions defined in SQL:1999

Scalar functions

Returns one value per invocation (one value per row)

Aggregate functions

Mathematical calculations such as max & min across rows

Window functions

Aggregate functions that operate on a subset of rows

Table functions

Return a set of rows for each input row – used to obtain information about Snowflake features

System functions

Control & information functions – usually SYSTEM\$... – SYSTEM\$CANCEL_ALL_QUERIES

```
SELECT SYSTEM$TYPEOF('abc');
```

SYSTEM\$TYPEOF('ABC')

1

VARCHAR(3)[LOB]





Functions

Supports most standard SQL functions defined in SQL:1999

Scalar functions

Returns one value per invocation (one value per row)

Aggregate functions

Mathematical calculations such as max & min across rows

Window functions

Aggregate functions that operate on a subset of rows

Table functions

Return a set of rows for each input row – used to obtain information about Snowflake features

System functions

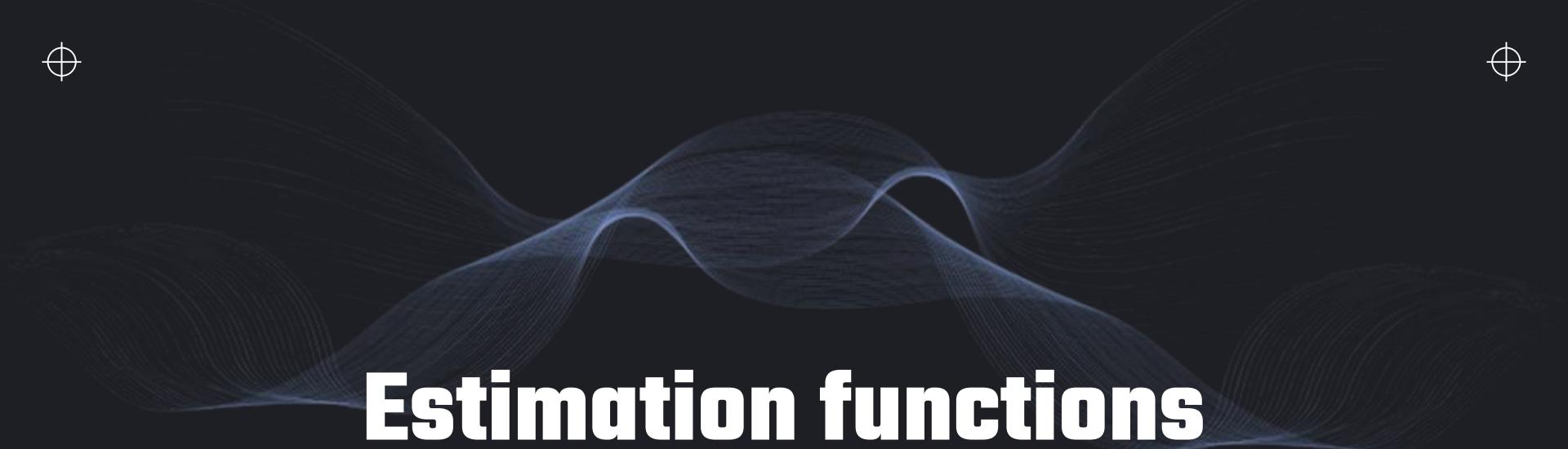
Control & information functions – usually SYSTEM\$... – SYSTEM\$CANCEL_ALL_QUERIES

UDFs + External

Define functions – store & execute outside of Snowflake

<https://docs.snowflake.com/en/sql-reference/intro-summary-operators-functions>





Estimation functions





Estimating functions

The idea:

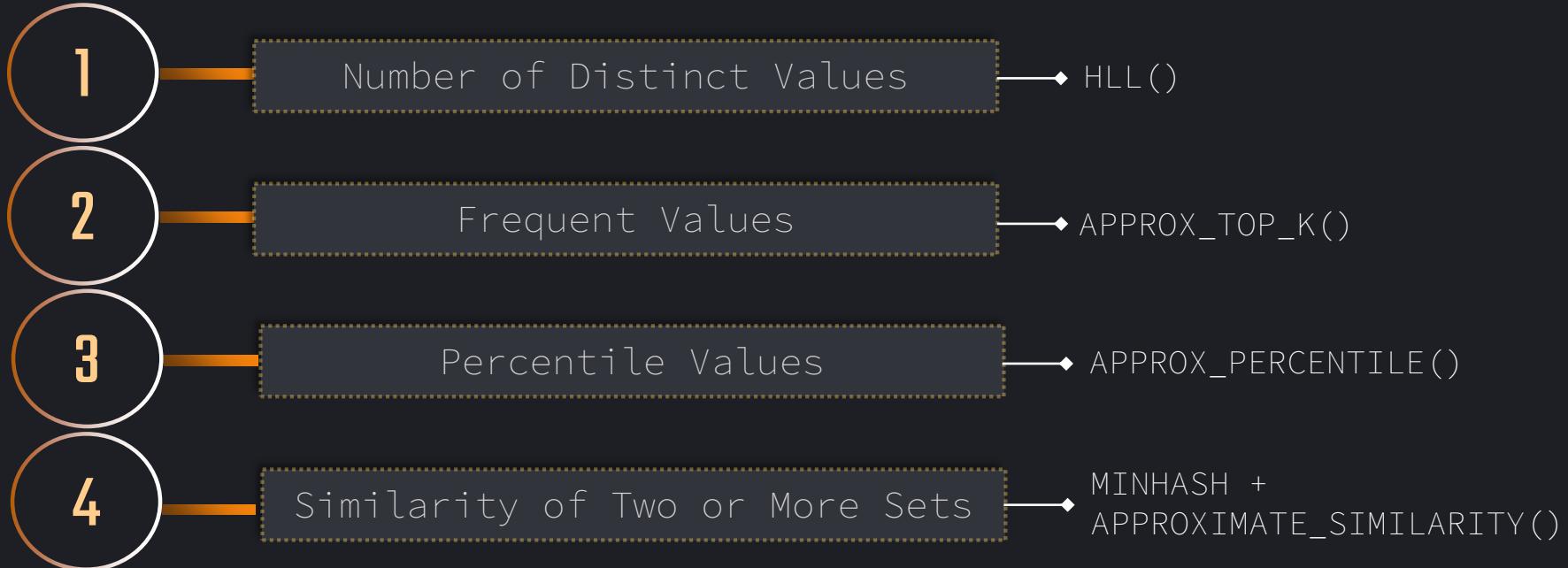
Exact calculations on very large tables can be very compute-/ memory-intensive.

Mathematical algorithms can approximate the exact number, they might be good enough and require fewer resources.



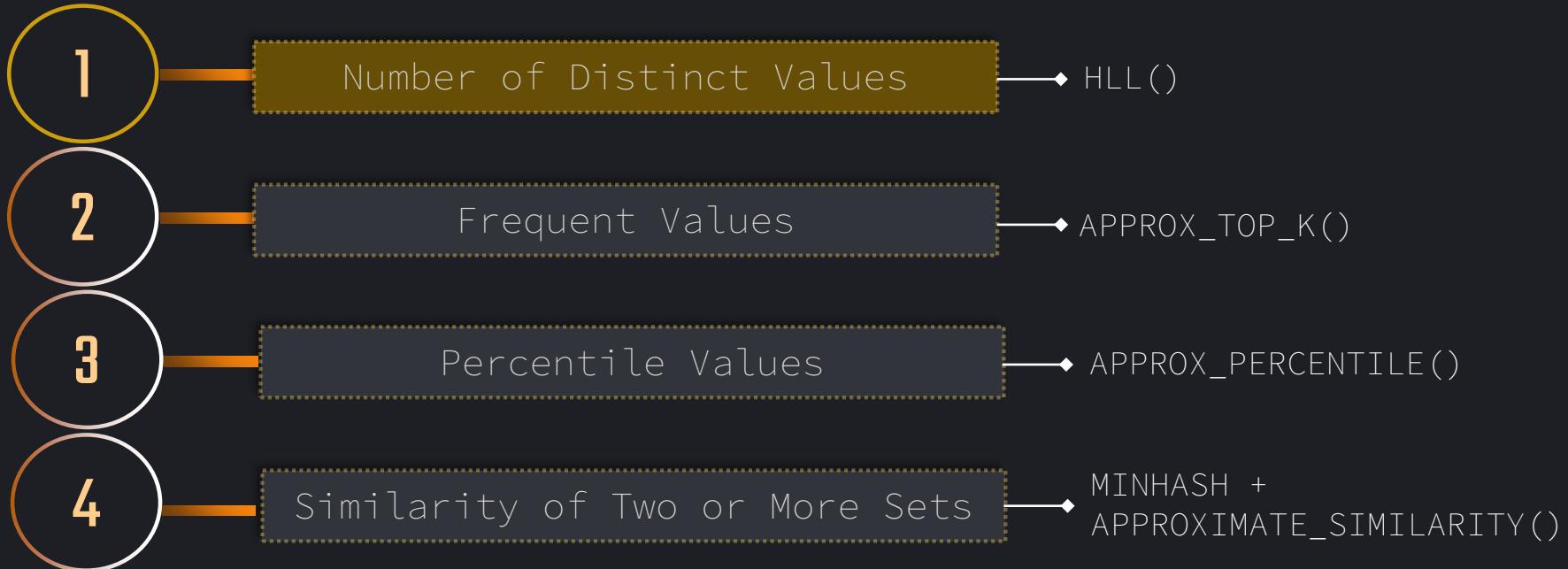


Estimating functions





Estimating functions





Number of Distinct Values

The idea:

HyperLogLog (cardinality estimation algorithm) is used to estimate the number of distinct values.

Situation:

COUNT(DISTINCT (COLUMN1,...))



Large input

Average error is acceptable

1.62338%

HLL(COLUMN1,...)

APPROX_COUNT_DISTINCT (COLUMN1,...)



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Number of Distinct Values

CUSTOMER

123 C_CUSTKEY

Aa C_NAME

Aa C_ADDRESS

123 C_NATIONKEY

NUMBER(38,0)

Aa C_PHONE

VARCHAR(15)

123 C_ACCTBAL

NUMBER(12,2)

Aa C_MKTSEGMENT

VARCHAR(10)

Aa C_COMMENT

VARCHAR(117)

```
SELECT HLL(C_NAME) FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1000.CUSTOMER;
```

HLL(C_NAME)

148,133,819

-1.244%

Query Details

Query duration

...

5.6s

Rows

1

```
SELECT COUNT(DISTINCT C_NAME) FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1000.CUSTOMER;
```

COUNT(DISTINCT(C_NAME))

150,000,000

Query Details

Query duration

...

12s

Rows

1

003-1040559

1250 003-77156.8

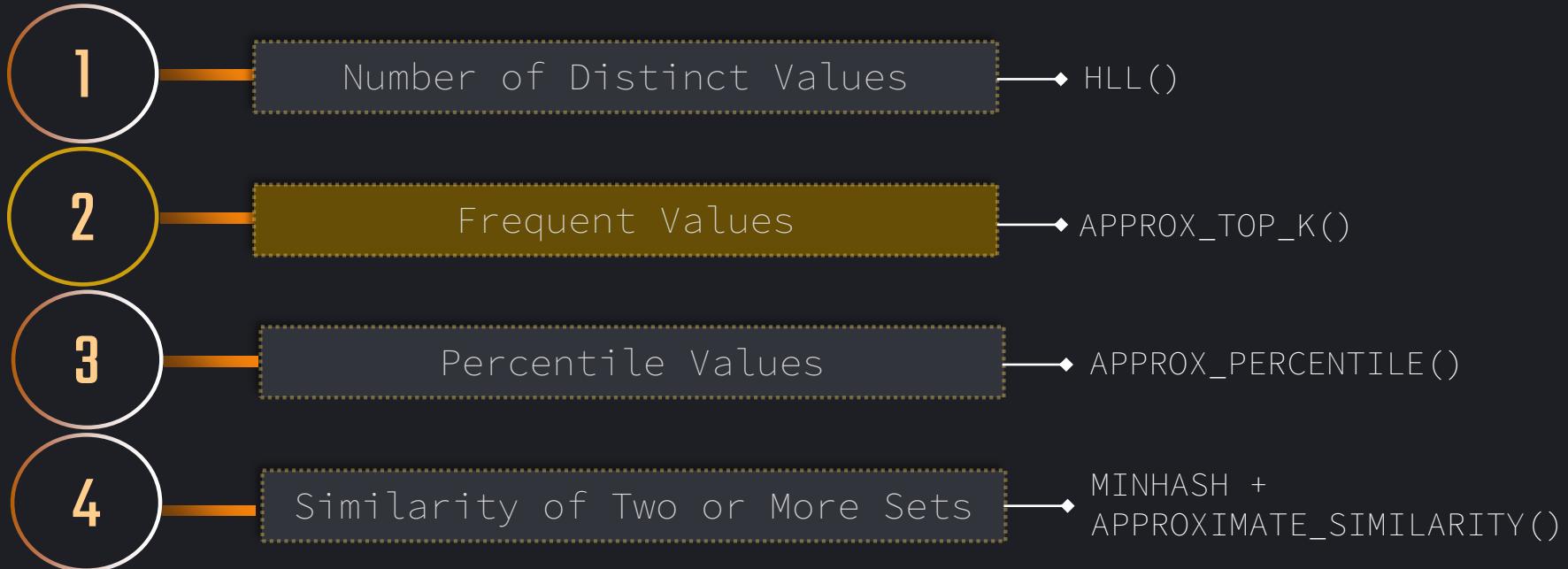
1760 0009-14563.7

73273





Estimating functions





Frequent Values

The idea:

Space-Saving algorithm is used to estimate the most frequent values along with their frequency.

Function:

```
APPROX_TOP_K (COLUMN)
```



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Frequent Values

APPROX_TOP_K (COLUMN)

APPROX_TOP_K (COLUMN, $<k>$)

APPROX_TOP_K (COLUMN, $<k>$, $<counters>$)

$k=1$

k = No. of values whose frequency should be approximated

$counters$ = Max. no. of distinct values that can be tracked

$count >> k$

$count$ large
⇒ more accurate



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Frequent Values

SNOWFLAKE_SAMPLE_DATA.TPCDS_SF10TCL.STORE_SALES 28.8B rows

```
SELECT SS_CUSTOMER_SK, CO  
FROM STORE_SALES  
GROUP BY SS_CUSTOMER_SK
```

SS_CUSTOMER_SK	COUNT(SS_CUSTOMER_SK)
64,714,499	360
11,024,612	373
51,885,590	800
21,882,280	800

Query Details

...

Query duration

17m 43s

Rows

65.0M

```
SELECT APPROX_TOP_K (SS_CUSTOMER_SK,5,20)  
FROM STORE_SALES
```

APPROX_TOP_K (SS_CUSTOMER_SK,5,20)

1 [[20952969, 174160540], [44412221, 174160540], [30221701, 174160530], [30594069, 174160530], [53273150, 174160530]]

Query Details

...

Query duration

2m 59s

Rows

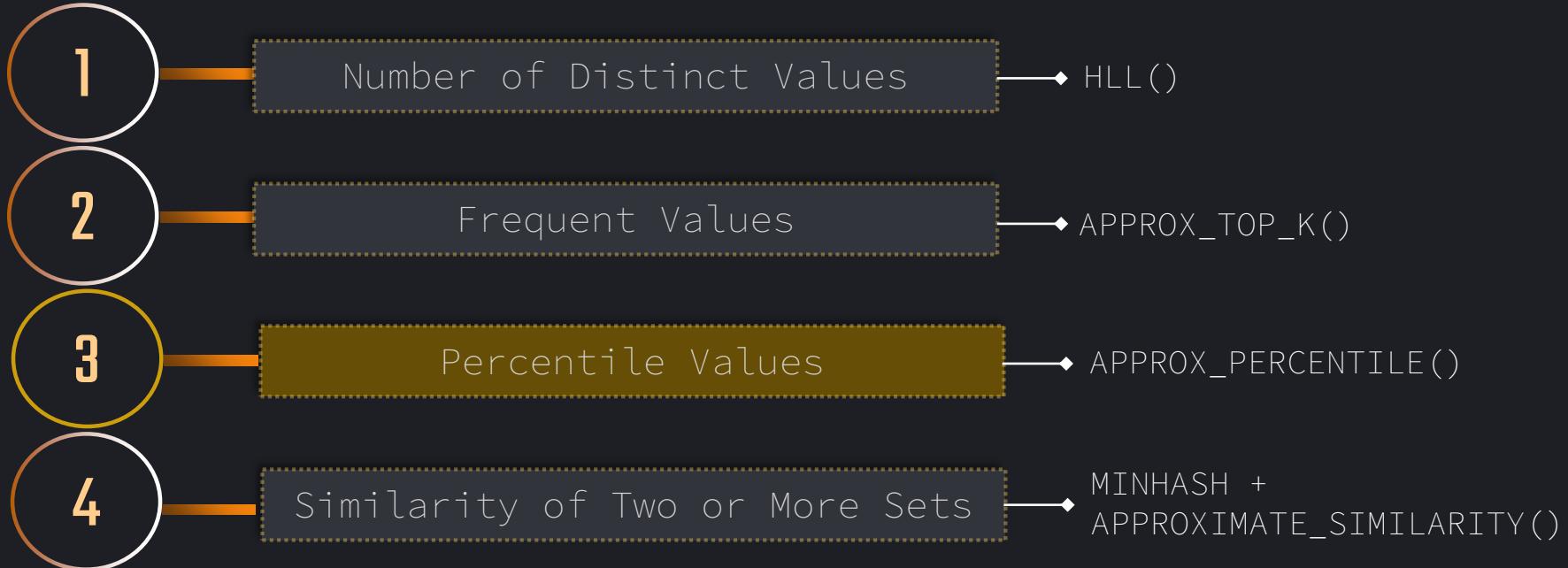
1

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273





Estimating functions





Percentile Values

The idea:

t-Digest algorithm is used to estimate percentile values.

Function:

```
APPROX_PERCENTILE (COLUMN,<percentile>)
```

→ Returns the percentile value





Percentile Values

SNOWFLAKE_SAMPLE_DATA.TPCH_SF1000

1.5B rows

```
SELECT APPROX_PERCENTILE(O_TOTALPRICE, 0.5)  
FROM ORDERS;
```

APPROX_PERCENTILE(O_TOTALPRICE,0.5)
144,288.43314293

Query Details

...

Query duration

20s

Rows

1

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY O_TOTALPRICE)  
FROM ORDERS;
```

PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY O_TOTALPRICE)
144,288.015

Query Details

...

Query duration

1m 26s

Rows

1



003-1040559

1250 003-77156.8

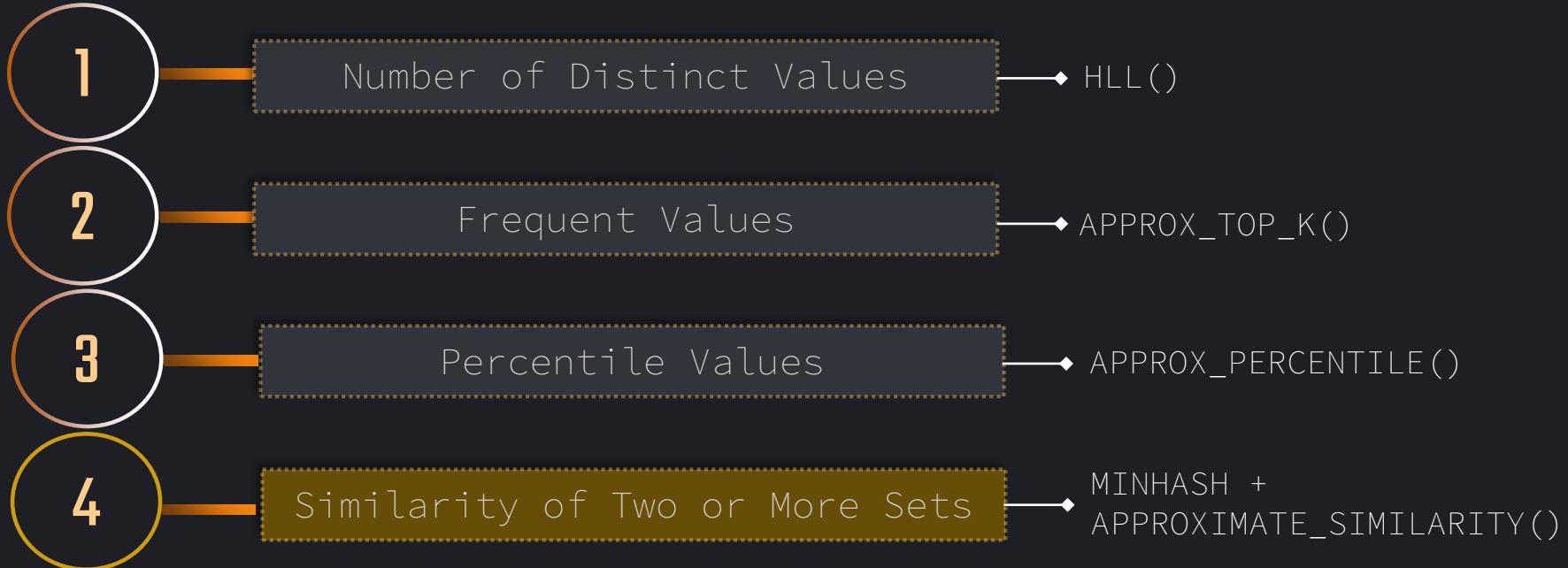
1760 0009-14563.7

73273

MOM2



Estimating functions



Similarity of Two or More Sets

The idea:

Uses MinHash to estimate the similarity between two or more data sets.

Typically the Jaccard similarity coefficient is used to compare similarity.

$$J(A, B) = (A \cap B) / (A \cup B)$$

MinHash can estimate $J(A, B)$ quickly.



Computationally expensive!





Similarity of Two or More Sets

2-step-process

1

MINHASH

```
SELECT MINHASH(100, *) AS mh FROM mhtab1;  
SELECT MINHASH(100, *) AS mh FROM mhtab2;
```

Returns a MinHash state

```
SELECT MINHASH(7, O_ORDERKEY) AS mh FROM ORDERS;
```

```
{ [  
MH  
{  
  "state": [  
    2200169610250,  
    22818457966550,  
    2507497641893,  
    12337014946743,  
    5083517324927,  
    1039435359430,  
    967271249674  
  ],  
  "type": "minhash",  
  "version": 1  
}
```



Similarity of Two or More Sets

2-step-process

1

MINHASH

```
SELECT MINHASH(100, *) AS mh FROM mhtab1;  
SELECT MINHASH(100, *) AS mh FROM mhtab2;
```

Returns a MinHash state

2

APPROXIMATE_SIMILARITY

```
SELECT APPROXIMATE_SIMILARITY(mh)  
FROM  
((SELECT MINHASH(100, *) AS mh FROM mhtab1)  
UNION ALL  
(SELECT MINHASH(100, *) AS mh FROM mhtab2));
```

k is # of hash functions
the larger k, the more accurate

Use MinHash states to calculate
similarity with APPROXIMATE_SIMILARITY().

MH

```
{ "state": [ 2200169610250, 22818457966550, 682837  
{ "state": [ 40568503727715, 129714795219004, 4095
```

(SELECT MINHASH(100, *) AS mh FROM mhtab1) UNION ALL

UNION ALL

Similarity of Two or More Sets

2-step-process

1

MINHASH

```
SELECT MINHASH(100, *) AS mh FROM mhtab1;  
SELECT MINHASH(100, *) AS mh FROM mhtab2;
```

Returns a MinHash state

2

APPROXIMATE_SIMILARITY

```
SELECT APPROXIMATE_SIMILARITY(mh)  
FROM  
((SELECT MINHASH(100, *) AS mh FROM mhtab1)  
UNION ALL  
(SELECT MINHASH(100, *) AS mh FROM mhtab2)),
```

Use MinHash states to calculate similarity with APPROXIMATE_SIMILARITY().

Returns a value between 0 and 1.

APPROXIMATE_SIMILARITY(MH)

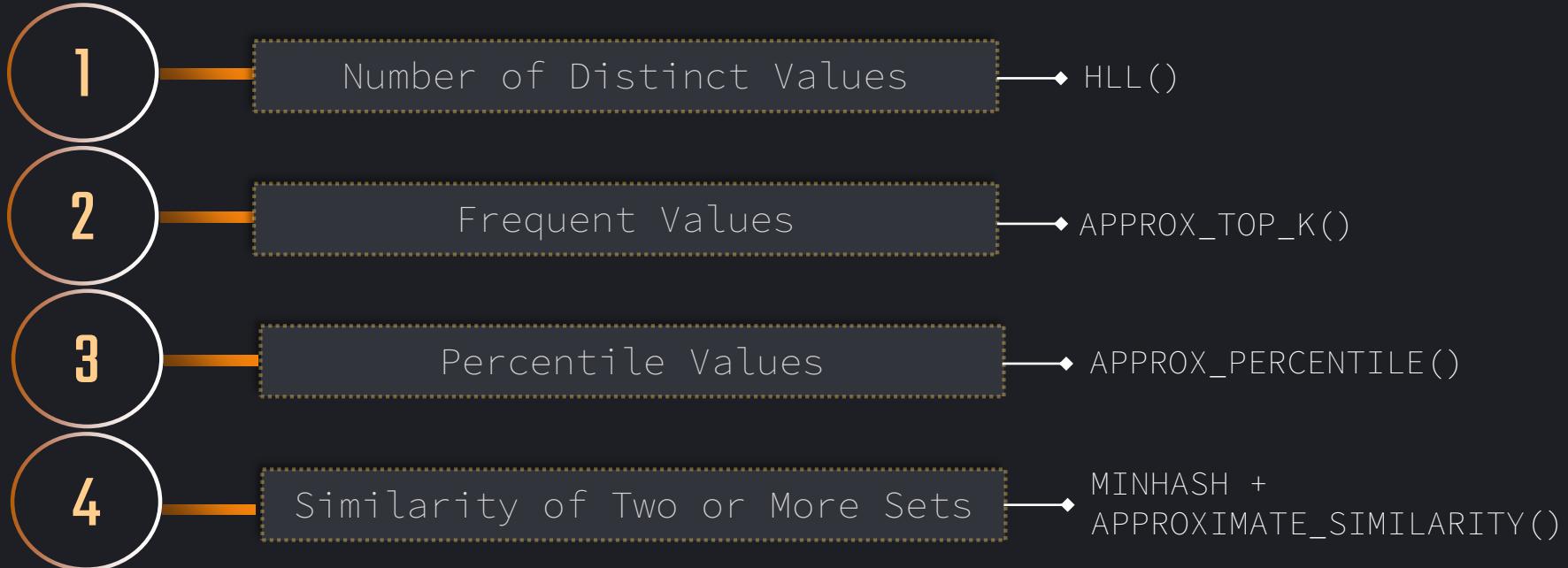
1	0.01
---	------

(SELECT MINHASH(100, *) AS mh FROM mhtab1) UNION ALL (SELECT MINHASH(100, *) AS mh FROM mhtab2))

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273



Estimating functions





User-defined functions (UDFs)

Way of extending functionality with additional functions.

Supported languages:

- o SQL
- o Python
- o Java
- o JavaScript

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

	status	...
1	Function ADD_TWO successfully created.	

```
select add_two(3);
```

ADD_TWO(3)	
1	5





User-defined functions (UDFs)

Way of extending functionality with additional functions.

Supported languages:

- o SQL
- o Python
- o Java
- o JavaScript

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





User-defined functions



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

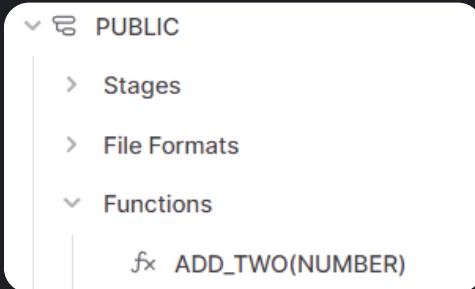
User-defined functions (UDFs)

Scalar functions

Returns one output row per input row

Tabular functions

Returns a tabular value for each input row



Securable schema-level object



External Functions



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



External functions

User-defined functions that are stored and executed outside of Snowflake.

Calls code that is executed outside of Snowflake.

- ✓ No code is stored in function definition
- ✓ Reference third-party libraries, services and data

```
create external function my_az_funct(string_col VARCHAR)
returns variant
api_integration = azure_external_api_integration
as 'https://my-api-management-svc.azure-api.net/my-api-url/my_http_function'
```





External functions

User-defined functions that are stored and executed outside of Snowflake.

Calls code that is executed outside of Snowflake.

- ✓ No code is stored in function definition
- ✓ Reference third-party libraries, services and data
- ✓ Remotely executed code ⇒ "remote service"
- ✓ Security related information are stored in API integration
- ✓ Schema-level object





External functions

User-defined functions that are stored and executed outside of Snowflake.

Calls code that is executed outside of Snowflake.

Examples:

- o AWS Lambda function
- o Microsoft Azure function
- o HTTPS server





Advantages & Limitations

Advantages:

- ✓ Additional languages including GO and C#
- ✓ Accessing 3rd-party libraries such as machine learning scoring libraries
- ✓ Can be called from Snowflake and from other software

Limitations:

- Must be scalar
- Slower performance (overhead + fewer optimizations)
- Not sharable





Stored Procedures



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Stored Procedures

Way of extending functionality with additional functions.

Stored Procedure

Typically performs database operations - usually administrative operations like DELETE, UPDATE or INSERT.

Doesn't need to return a value

Caller or Owner's rights

UDF

Typically calculate and return a value.

Need to return a value

No need to have access to objects reference in the function.





Stored Procedures

Way of extending functionality with additional functions.

Supported languages:

- Snowflake Scripting
(Snowflake SQL + procedural logic)
- JavaScript
- Snowpark API
(Python, Scala, Java)





Stored Procedures

Creating a procedure

```
create procedure find_min(n1 int, n2 int)
returns int
language sql
    as
BEGIN
    IF (n1 < n2)
        THEN RETURN n1;
        ELSE RETURN n2;
END IF;
END;
```

Securable objects

- > File Formats
- > Functions
- ▼ Procedures
 - ↳ CONCAT_TEXT(VARCHAR, ...)
 - ↳ UPDATE_FUNCTION()
 - ↳ UPDATE_FUNCTION(NUMB...





Stored Procedures

Calling a procedure

```
call find_min(5, 7);
```

Result

FIND_MIN	
1	5





Stored Procedures

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;
END;
```



END;
SET test_col = 3;
SELECT * FROM test1;



Stored Procedures

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;
```

	UPDATE_TEST_TABLE	...
1	null	



END:

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Stored Procedures

If argument is used in SQL statement ⇒ use :argument

```
create procedure update_test_table(new_value varchar)
returns int
language sql
as
BEGIN
UPDATE manage_db.public.test1
SET test_col = :new_value;
END;
```



END;
003-1040559 1250 003-77156.8 1760 0009-14563.7 73273
SET test_col = :new_value;
playground@localhost:~\$



Stored Procedures

If argument is used in SQL statement to refer to as an object
⇒ use IDENTIFIER(:argument)

```
create procedure update_table(new_v varchar,table_name varchar)
returns varchar
language sql
as
BEGIN
    UPDATE IDENTIFIER(:table_name)
    SET test_col = :new_value;
    RETURN 'Successfully updated table';
END;
```



END:

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

KELOKI-200003310533 abased.cspsc }



Stored Procedures

Calling a stored procedure

```
call update_table('new_value', 'table_name');
```





Privileges

Runs either with caller's or owner's rights

Runs with the
caller's privileges

Runs with the
owner's privileges

Can make user of user information
(e.g session variables)

Delegation of
Administrative tasks





Stored Procedures

When creating specify rights
⇒ use execute as caller/owner

DEFAULT
OWNER

```
create procedure update_table(new_v varchar,table_name varchar)
execute as caller
returns int
language sql
as
BEGIN
UPDATE IDENTIFIER(:table_name)
SET test_col = :new_value;
END;
```



END:
003-1040559 1250 003-77156.8 1760 0009-14563.7 73273
SET test_col = :new_value;



Secure UDFs & Stored Procedures



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Secure UDFs & Stored Procedures

```
DESC FUNCTION MANAGE_DB.PUBLIC.ADD_TWO(NUMBER);
```

	property	value
1	signature	(N NUMBER)
2	returns	NUMBER(38,0)
3	language	SQL
4	body	n+2

↳ pqql ↳ n+2

Secure UDFs & Stored Procedures

- ✓ Hide certain information, e.g. definition
- ✓ Prevent users from seeing underlying data

```
CREATE SECURE FUNCTION <function_name> ...
```





Secure UDFs & Stored Procedures

Trade-off:

Reduced query performance \Leftrightarrow Security

Use-case:

- Consider purpose of UDF / Stored Procedure
- NOT make it secure if it is define just for convenience
- Make it secure when the data is sensitive enough





Sequences

```
CREATE SEQUENCE my_seq  
START = 1  
INCREMENT = 1; } DEFAULT = 1
```

Sequences

MY_SEQ

```
SELECT my_seq.nextval;
```

NEXTVAL

1

NEXTVAL

2

Sequences

```
SELECT my_seq.nextval;
```

- ✓ Are securable objects
- ✓ Typically create for default values





Sequences



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Sequences

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

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

Sequences

- ✓ Not guaranteed to be gap-free

ID	FIRST_NAME
1	Maria;
2	Frank





Semi-structured data



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Semi-structured data

What is semi-structured data?

- no fixed schema
- contains tags/labels and a nested structure

JSON

```
{  
  "courses": [  
    {  
      "topic": "Snowflake",  
      "level": "All levels"  
    },  
    {  
      "topic": "SQL",  
      "language": ["English", "German"]  
    },  
    {  
      "topic": "Azure",  
      "level": "Beginner"  
    }  
  ]  
}
```

What is structured data?

Data has a well-defined structure

	Loan_ID	loan_status	Principal	terms	effective_date
1	xqd20168902	PAIDOFF	1000	30	9/8/2016
2	xqd20160003	PAIDOFF	1000	30	9/8/2016
3	xqd20160005	PAIDOFF	1000	30	9/9/2016
4	xqd20160006	PAIDOFF	1000	30	9/9/2016





Semi-structured Data Types

Supported formats:

- JSON
- XML
- PARQUET
- ORC
- Avro





Data Types in Snowflake

OBJECT

Unordered set of name value pairs.

```
[  
    "topic":"Snowflake",  
    "level":"All levels"  
,
```

ARRAY

Consists of 0 or more pieces of data.

```
["USA", "India", "Canada"]
```



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Data Types in Snowflake

VARIANT

Can store values of any other data type including ARRAY and OBJECT.

Use-cases

Explicitely define hierarchy of ARRAYS and OBJECTs

Suitable to store and query semi-structured data.

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}
```





Data Types in Snowflake

VARIANT

SQL nulls are just stored as "null" strings

They are called JSON null (or "VARIANT null")

Non-native strings (e.g. dates) are stored in strings

Maximum length is 16 MB.

(uncompressed per row)

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}
```



Loading semi-structured data

VARIANT

"Native support for semi-structured data"

Load the data as it is and transform it later.

ELT approach

Extract & Load raw data

Analyze / Parse

Flatten

VARIANT

COPY INTO ...



Query

Semi-structured data

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

Query semi-structured data

To access elements of VARIANT column use :

```
SELECT raw_column:courses  
FROM variant_table
```

RAW_COLUMN:COURSES

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



[[RAW_COLUMN:COURSES

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

BUOMIJSKO:

```
[  
    003-1040559, 1250, 003-77156.8, 1760, 0009-14563.7, 73273 ]
```

```
[  
    003-1040559, 1250, 003-77156.8, 1760, 0009-14563.7, 73273 ]
```

Query semi-structured data

To access elements of VARIANT column use :

```
SELECT $1:courses  
FROM variant_table
```

RAW_COLUMN:COURSES

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



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

[[RAW_COLUMN:COURSES

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

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273]



Query semi-structured data

To access elements of VARIANT column use :
Use . to access subsequent elements

```
SELECT column_name:courses.snowflake  
FROM variant_table
```

```
{"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"  
                ]  
            }  
        }  
    }  
}
```

```
[[ RAW_COLUMN:COURSES.SNOWFLAKE  
{  
    "module1": {  
        "topic": "Introduction",  
        "formats": [  
            "video lectures",  
            "hands-on",  
            "quizzes"  
        ]  
    },  
    "module2": {  
        "topic": "Introduction",  
        "formats": [  
            "video lectures",  
            "hands-on",  
            "quizzes"  
        ]  
    }  
}
```

```
RAW_COLUMN:COURSES.SNOWFLAKE  
...  
1 { "module1": { "topic": "Introduction", "formats": [ "video lectures" ] }, "module2": { "topic": "Introduction", "formats": [ "video lectures", "hands-on", "quizzes" ] } }
```



```
003-1040559 1250 003-77156.8 1760 0009-14563.7 73273 ]
```



Query semi-structured data

To access elements of VARIANT column use :
Use [] to access array elements

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

```
{"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"
        ]
    }
}}
```

RAW_COLUMN:COURSES.AZURE.MODU	
1	"hands-on"

Query Semi-structured data

To access elements of VARIANT column use :
Use [] to access array elements

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

	RAW_COLUMN:COURSES.SNOWFLAKE MODULE2.FORMATS[0]::
1	video lectures

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





Flatten hierarchical data



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Flatten data

```
FLATTEN(INPUT => <expression> )
```

Used to convert semi-structured data into relational table view

```
SELECT * FROM TABLE(FLATTEN(INPUT => [2,4,6]))
```

Cannot be used in COPY command!

	SEQ	KEY	PATH	INDEX	VALUE	THIS
1	1	null	[0]	0	2	[2, 4, 6]
2	1	null	[1]	1	4	[2, 4, 6]
3	1	null	[2]	2	6	[2, 4, 6]

Produces a lateral view:

Contains references to other tables in FROM clause





Unstructured data



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Unstructured data support

What is unstructured data?

- Does not fit into any pre-defined data model

- video files



- audio files



- documents



Snowflake supports:

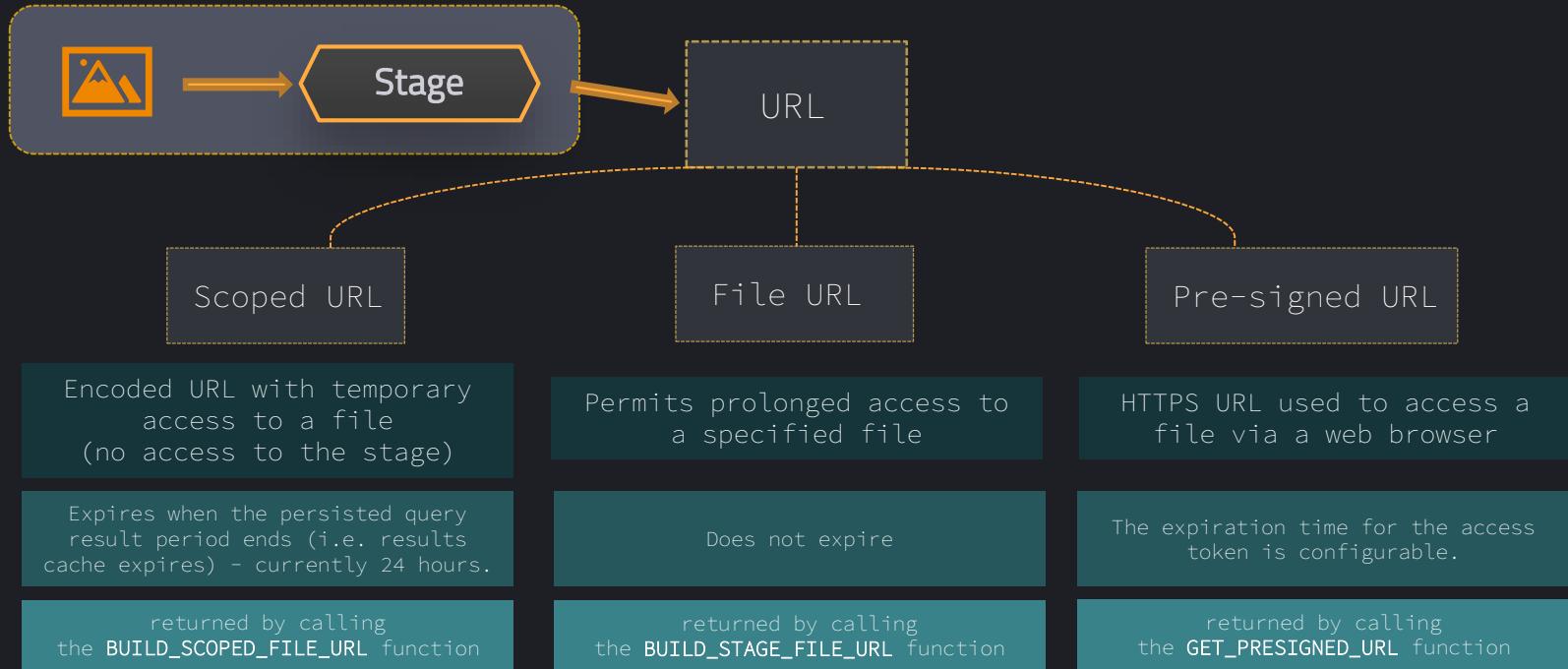
- Access files through URL in cloud storage.
 - Share file access URLs.

Internal & External Stages



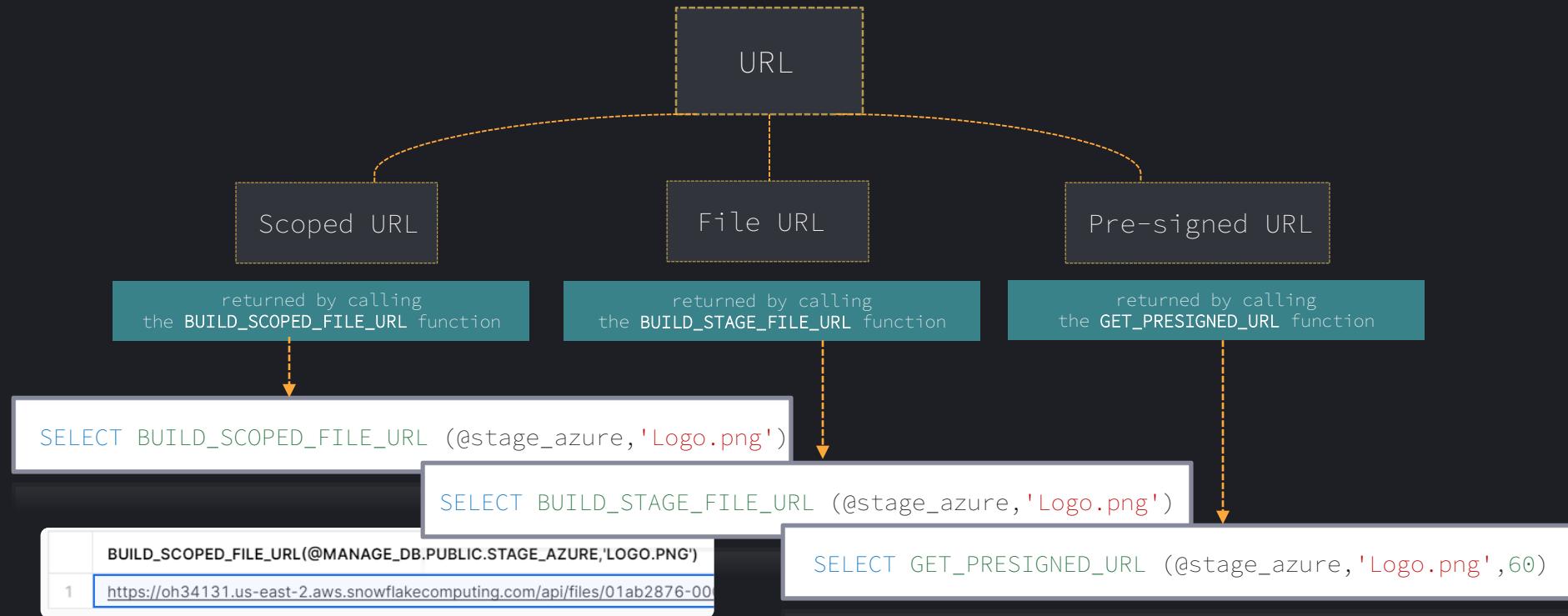


Unstructured data support

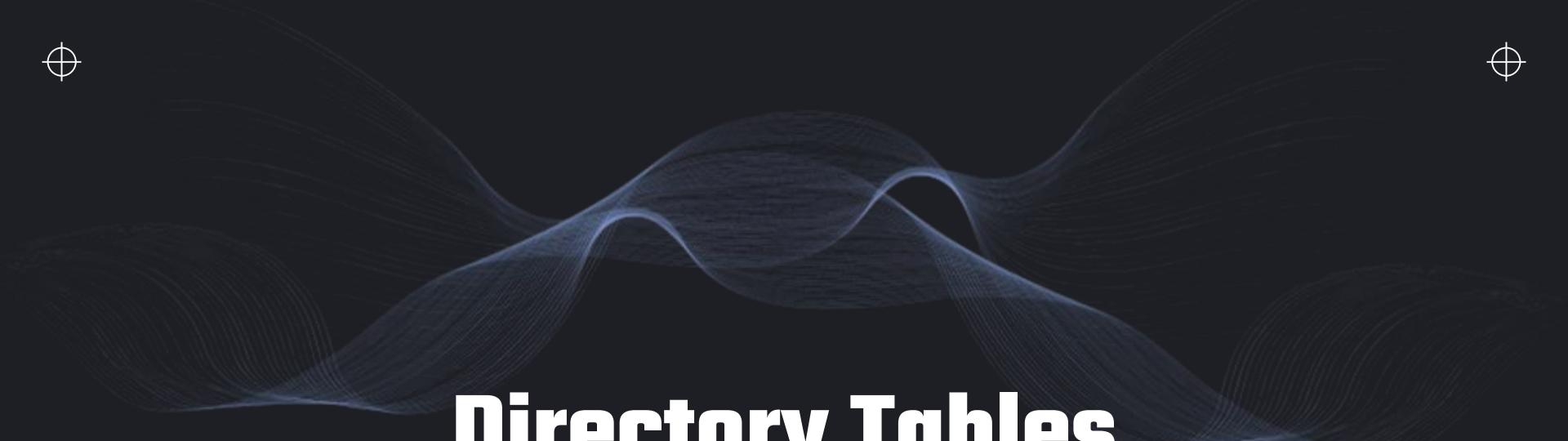




SQL File Functions



003-1040559 1250 003-77156.8 1760 0009-14563.7 73273



Directory Tables

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Directory Tables

What is a directory table?

Stores metadata of staged files

- Layered on a stage
- Can be queried with sufficient privileges (on stage)
- Retrieve file URLs to access files

Needs to be enabled for stages

```
CREATE STAGE stage_azure  
URL = <'url'>  
STORAGE_INTEGRATION = integration  
DIRECTORY = (ENABLE = TRUE)
```

DEFAULT
FALSE

```
ALTER STAGE stage_azure  
SET DIRECTORY = (ENABLE = TRUE)
```





Directory Tables

```
CREATE STAGE stage_azure  
URL = <'url'>  
STORAGE_INTEGRATION = integration  
DIRECTORY = (ENABLE = TRUE)
```

```
SELECT * FROM DIRECTORY(@stage_azure)
```

```
ALTER STAGE stage_azure REFRESH;
```

Manual refresh

Automatical refresh
using event notification

Scoped URL

BUILD_SCOPED_FILE_URL function

RELATIVE_PATH	SIZE	LAST_MODIFIED	MD5	ETAG	FILE_URL
Query produced no results					
Logo.png	46,603	2019-22.000 -0700	e24ca582347f57f4	"0x8DB2C513	https://oh34131.us-eas...
myfile_0_0_0.csv.gz	33,785	2018-52.000 -0700	a3169775b4e20a3	"0x8DB29241	https://oh34131.us-eas...

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





Data Sampling



003-1040559 1250 003-77156.8

1760 0009-14563.7 73273





Data Sampling

Why Sampling?



10 TB



500 GB





Data Sampling

Why Sampling?

- Use-cases: Query development, data analysis etc.
- Faster & more cost efficient (less compute resources)





Data Sampling Methods

```
SELECT * FROM table  
SAMPLE ROW (<p>) SEED(15)
```

Percentage of rows

Reproducible results

ROW or BERNOUILLI method

```
SELECT * FROM table  
SAMPLE SYSTEM (<p>) SEED(15)
```

BLOCK or SYSTEM method





Data Sampling Methods

ROW or BERNOULLI method

Every row is chosen with percentage p

More "randomness"

Smaller tables

BLOCK or SYSTEM method

Every block is chosen with percentage p

More effective processing

Larger tables





Tasks



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Tasks

Used to schedule execution of SQL statement / stored procedures

Often combined with streams to set up continuous ETL workflows

Run using privileges of task OWNER

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

Schema-level object

Can be cloned

EXECUTE MANAGED TASK

Account

CREATE TASK

Schema

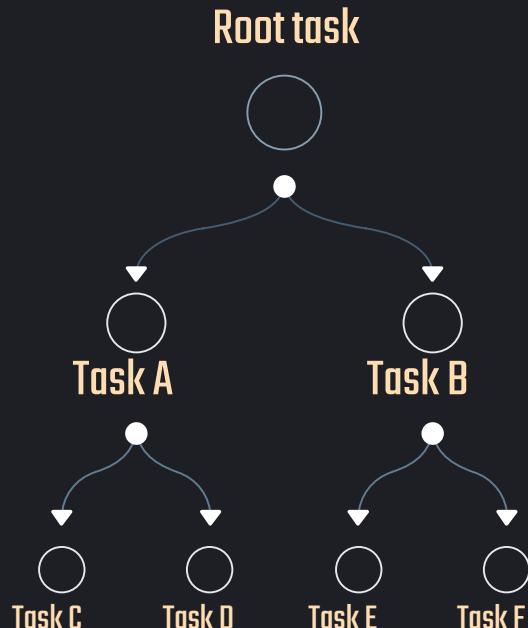
USAGE

Warehouse

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



Directed Acyclic Graph (DAG)



Directed Acyclic Graph (DAG)

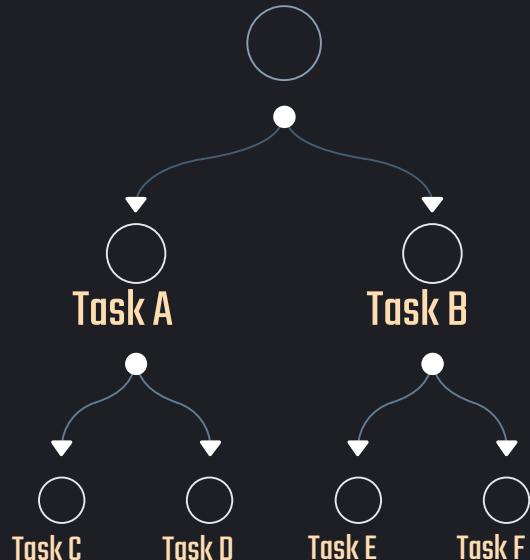
Limited to

- 1000 tasks in total
- 100 child tasks



Tree of Tasks

Root task



Limited to

- 1000 tasks in total
- 100 child tasks

```
CREATE TASK my_task  
WAREHOUSE = my_wh  
AFTER my_task_a  
AS  
...;
```





Streams

Record (DML-) changes made to a table



Sales data



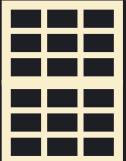
HR data

Data sources



Schema-level object

Can be cloned





Streams



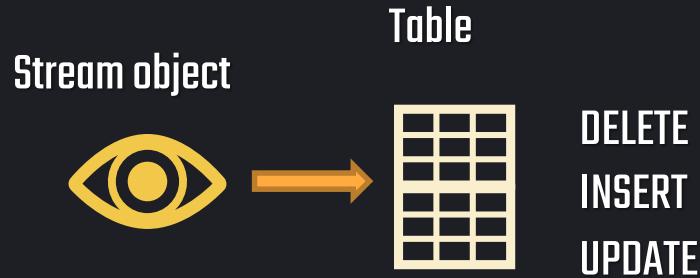
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Streams



Records (DML-)changes made to a table
This process is called change data capture (CDC)





Streams

Stream object

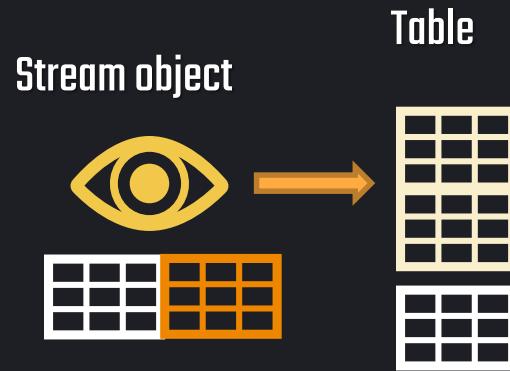


Table





Streams



METADATA\$ACTION
METADATA\$UPDATE
METADATA\$ROW_ID

METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
INSERT	FALSE	aed891912ed5d1978c29539dd
INSERT	FALSE	bb75b186692377775bcddf7121





Streams

```
CREATE STREAM my_stream  
ON TABLE my_table
```

Create Stream

```
SELECT * FROM my_stream
```

We can query from stream



003-1040559

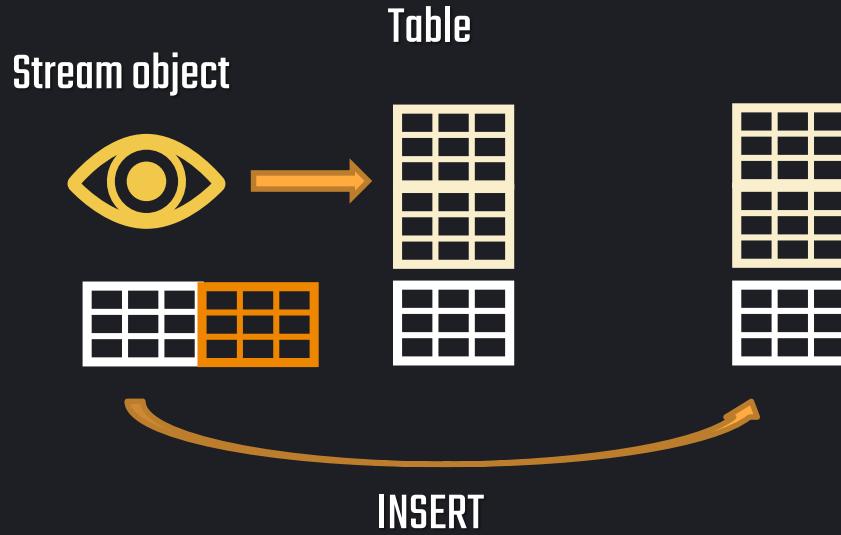
1250 003-77156.8

1760 0009-14563.7 73273



Streams

Consuming a stream

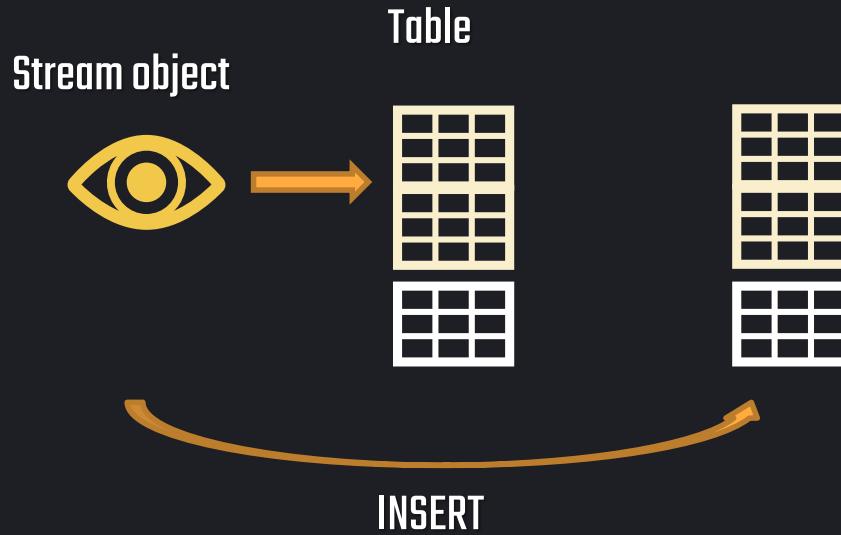




Streams

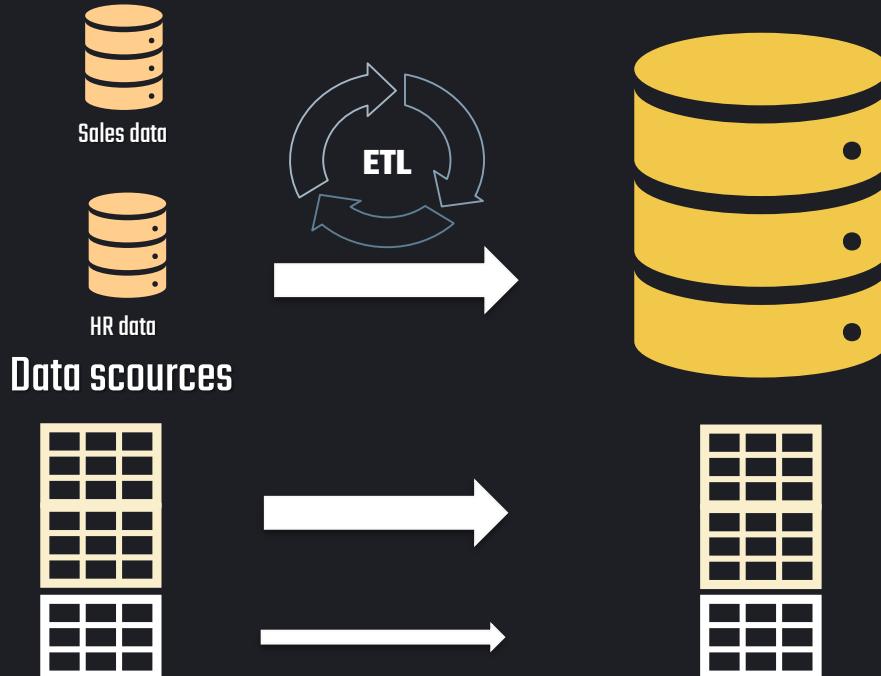
Consuming a stream

Empties records





Streams



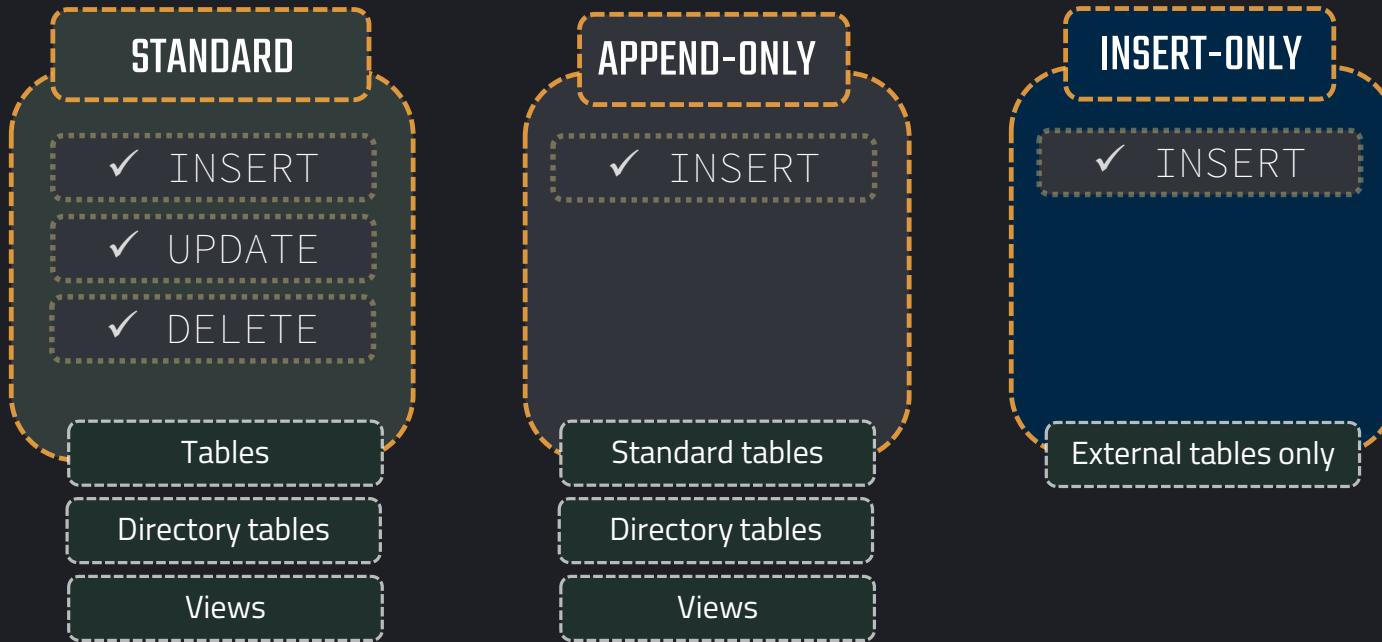
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Types of streams





Staleness

Stream becomes stale when offset is outside the data retention period of source table

`DATA_RETENTION_TIME_IN_DAYS`

Unconsumed change records won't be accessible anymore.

How frequently stream should be consumed

Column indicating when the stream is predicted to become stale

`DESCRIBE STREAM or SHOW STREAMS command`

`STALE_AFTER`

Stream extends retention to 14 days (default).

`DEFAULT=14`

Regardless of Snowflake edition

`MAX_DATA_EXTENSION_TIME_IN_DAYS`





Types of streams

STANDARD

- ✓ **INSERT**
- ✓ **UPDATE**
- ✓ **DELETE**

APPEND-ONLY

- ✓ **INSERT**



Combining Streams & Tasks

```
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);
```



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Additional Tools, Drivers & Connectors

Domain 1.0:
Snowflake Data Cloud Features & Architecture



01

02

03

04

05

06



01

02

03

04

05

06

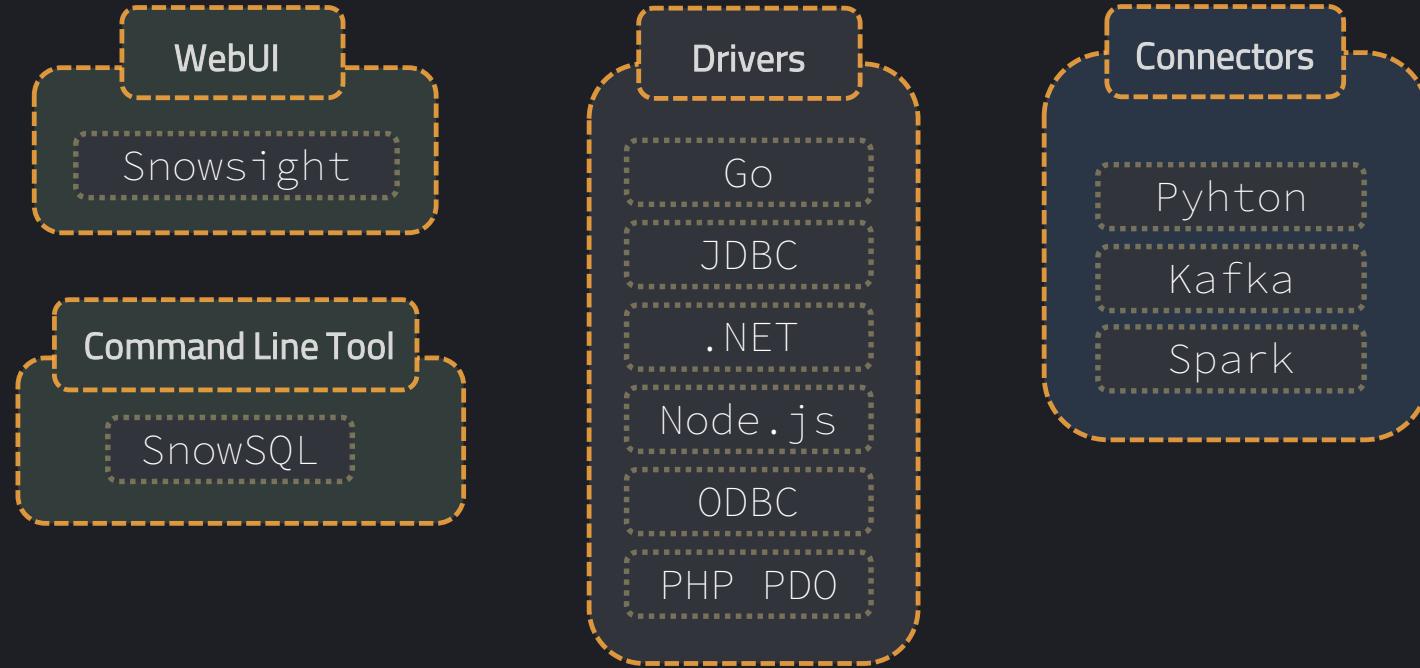


Transformations & Functions





Connectors & Drivers





Partner Connect

Create trial account and integrate them with Snowflake



Data Integration

Informatica

Talend

Matillion

Stitch

Qlik

Dataiku

Alteryx

DataRobot



ML & Data Science

CI/CD



Security & Governance



Business Intelligence

Immuta

Hunters

Alation

Sisense

Domo

Sigma





Snowflake Scripting



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Snowflake Scripting

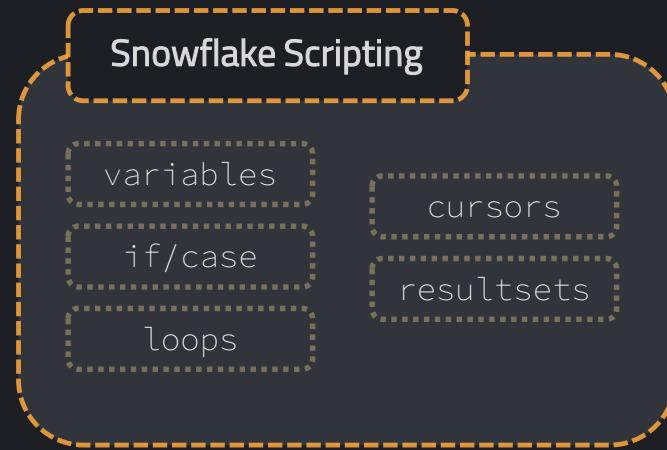
Extention to Snowflake SQL with added support for procedural logic



Most commonly



Outside of
stored procedures





Snowflake Scripting

Extention to Snowflake SQL with added support for procedural logic

Snowflake Scripting

if/case

IF/ELSE

CASE

loops

FOR

REPEAT

WHILE

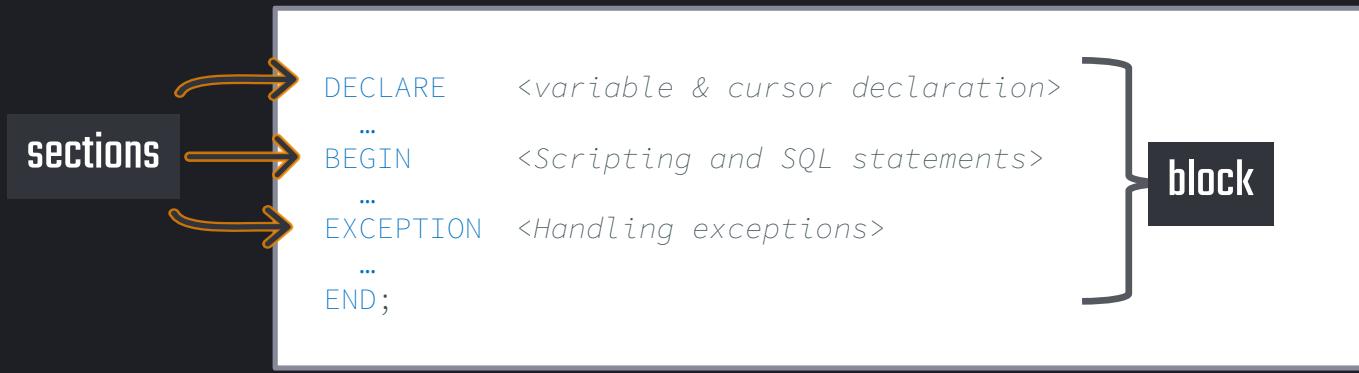
LOOP





Snowflake Scripting

How to write procedural code in a block





Snowflake Scripting

How to write procedural code in a block

optional

```
DECLARE      <variable & cursor declaration>  
...  
BEGIN        <Scripting and SQL statements>  
...  
EXCEPTION    <Handling exceptions>  
...  
END;
```

} block





Snowflake Scripting

How to write procedural code in a block

optional

```
DECLARE      <variable & cursor declaration>  
...  
BEGIN        <Scripting and SQL statements>  
...  
EXCEPTION    <Handling exceptions>  
...  
END;
```

} block





Snowflake Scripting

How to write procedural code in a block

Minimizing confusion

```
BEGIN  
    CREATE TABLE employee (id INTEGER,...);  
    CREATE TABLE store (id INTEGER, ...);  
END;
```

Object created in block can be used outside the block

} block





Snowflake Scripting

How to write procedural code in a block

```
CREATE PROCEDURE calc_area()
    RETURNS float
    LANGUAGE SQL
    AS
DECLARE
length_a float;
area float;
BEGIN
length_a := 4;
area := length_a * length_a;
RETURN area
END;
```

variables created in block can
be used only inside the block

Snowsight





Snowflake Scripting

How to write procedural code in a block

```
CREATE PROCEDURE calc_area()
    RETURNS float
    LANGUAGE SQL
    AS
$$
DECLARE
    length_a float;
    area float;
BEGIN
    length_a := 4;
    area := length_a * length_a;
    RETURN area
END;
$$
```

variables created in block can
be used only inside the block

Classic UI

SnowSQL





Snowpark



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Snowpark

Snowpark API provides support for three programming languages



Python Code

Build application and query data outside the system

No need to move data!

Converts to SQL

Snowflake

Process at scale with serverless Snowflake engine





Snowpark

Snowpark API provides support for three programming languages



Python



Java



Scala



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Snowpark

Lazy evaluation

Expression is not evaluated until it is needed

Pushdown

Code is pushed down to Snowflake and executed there

UDFs inline

Created functions can be executed in UDFs





Data Protection

Domain 6.0:
Data Protection and Data Sharing



01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100



Time Travel



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Data Protection Lifecycle

Current
Data Storage

Access and query data etc.

```
SELECT * FROM table
```

	ID	FIRST_NAME	LAST_NAME	EMAIL
1	1	Bourke	Treble	btreble0@g.co
2	2	Iormina	Lahy	ilahy1@sciencedaily.com
3	3	Tracy	Curwen	tcurwen2@spiegel.de
4	4	Megan	Omond	momond3@cyberchimps.com



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Data Protection Lifecycle

Current
Data Storage

Drop database or table accidentally?

```
DROP DATABASE prod_db;
```

Truncate or update table accidentally?

```
TRUNCATE TABLE prod_table;
```

	status
1	PROD_DB successfully dropped.

Time Travel enables accessing historical data.



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Time Travel

What is possible with Time Travel?

- Query deleted or updated data
- Restore tables, schemas and databases that have been dropped
- Create clones of tables, schemas and databases from previous state





Time Travel SQL

Query historic data within retention period.

```
SELECT * FROM table AT (TIMESTAMP => timestamp)
```

1

TIMESTAMP

```
SELECT * FROM table AT (OFFSET => -10*60)
```

2

OFFSET

```
SELECT * FROM table BEFORE (STATEMENT => query_id)
```

3

QUERY



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Time Travel SQL

Recover objects that have been dropped within retention period.

```
UNDROP TABLE table_name;
```

1

Table

```
UNDROP SCHEMA schema_name;
```

2

Schema

```
UNDROP DATABASE database_name;
```

3

Database





Considerations

UNDROP fails if an object with the same name already exists.

OWNERSHIP privileges are needed for an object to be restored.





Retention period



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Data Protection Lifecycle

Query historic data within retention period.

```
SELECT * FROM table AT (TIMESTAMP => timestamp)
```

1

TIMESTAMP

```
SELECT * FROM table AT (OFFSET => -10*60)
```

2

OFFSET

```
SELECT * FROM table BEFORE (STATEMENT => query_id)
```

3

QUERY





Retention period

Number of days for which this historical data is preserved
and Time Travel can be applied.

Configurable for
table, schema,
database and account

DATA_RETENTION_TIME_IN_DAYS = 2

DEFAULT = 1

For all accounts

Retention period of 0 "disables" time travel.





Retention period

Create table that overwrites
default retention period

```
CREATE TABLE table_name (
column1 int,
column2 varchar)
DATA_RETENTION_TIME_IN_DAYS = 0;
```

Alter table's retention period.

```
ALTER TABLE table_name (
SET DATA_RETENTION_TIME_IN_DAYS = 0;
```

```
ALTER ACCOUNT SET
DATA_RETENTION_TIME_IN_DAYS = 2;
```

Alter account's
default retention period

```
ALTER ACCOUNT SET
MIN_DATA_RETENTION_TIME_IN_DAYS = 2;
```

Set minimum
retention period





Retention period



Standard



Enterprise



Business Critical



Virtual Private

Time travel up to **1 day**

Time travel up to **90 days**

Time travel up to **90 days**

Time travel up to **90 days**





Fail Safe



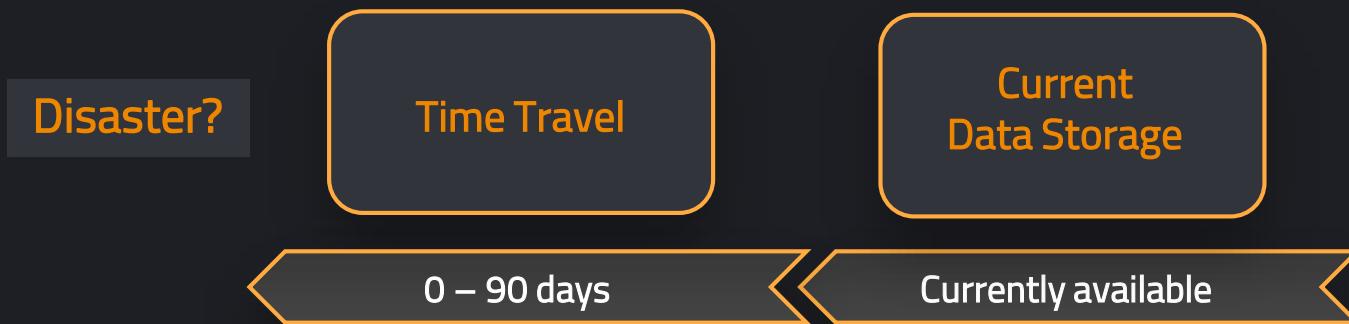
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Continuous Data Protection Lifecycle

- ✓ SELECT ... AT | BEFORE
- ✓ UNDROP
- ✓ Access and query data etc.



003-1040559

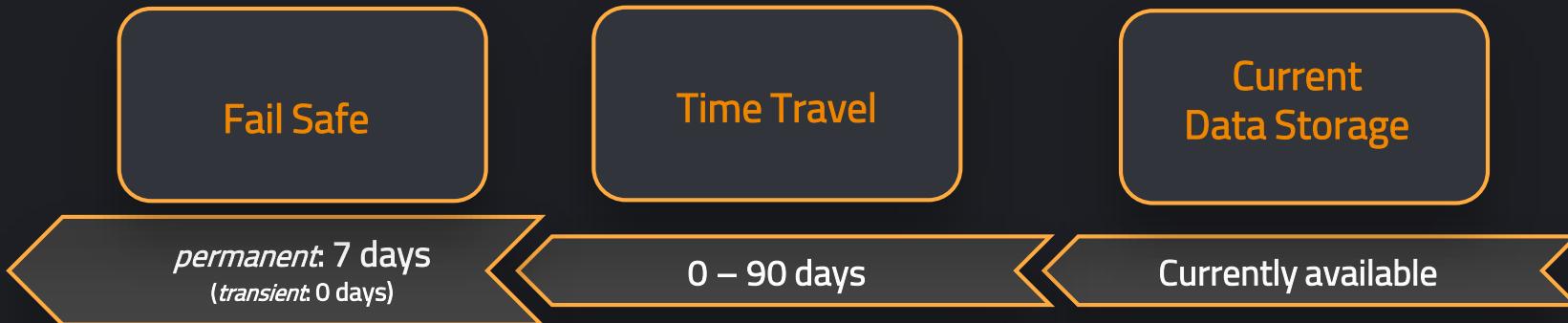
1250 003-77156.8

1760 0009-14563.7 73273



Continuous Data Protection Lifecycle

- ✓ Recovery beyond Time Travel
- ✓ Non-configurable
- ✓ No user operations/queries
- ✓ Restoring only by snowflake support
- ✓ SELECT ... AT | BEFORE
- ✓ UNDROP
- ✓ Access and query data etc.





Fail Safe

- ✓ Protection of historical data in case of disaster
- ✓ No user interaction & recoverable only by Snowflake
- ✓ Non-configurable 7-day period for permanent tables
- ✓ Period starts immediately after Time Travel period ends
- ✓ Contributes to storage cost





Table Types



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Table types

Permanent data

Only for data that does
not need to be protected

Non-permanent data

Permanent

Transient

Temporary

CREATE TABLE

CREATE TRANSIENT TABLE

CREATE TEMPORARY TABLE

Time Travel

0 – 90 days

0 or 1 day

0 or 1 day

Retention Period

Fail Safe

✓ Fail Safe

✗ Fail Safe

✗ Fail Safe

Auto-Drop

Until dropped

Until dropped

Only in session





Managing Storage Cost

Table types

Permanent data

Large tables that does
not need to be protected

Non-permanent data

Permanent

Transient

Temporary

CREATE TABLE

CREATE TRANSIENT TABLE

CREATE TEMPORARY TABLE

Time Travel

0 – 90 days

0 or 1 day

0 or 1 day

Retention Period

Fail Safe

✓ Fail Safe

✗ Fail Safe

✗ Fail Safe

Persistence

Until dropped

Until dropped

Only in session

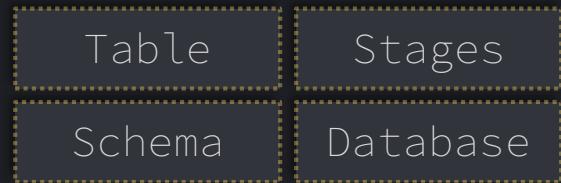




Table types notes

- ✓ Types are also available for other database objects

If database is transient all included objects are transient.



- ✓ For temporary table no naming conflicts with permanent/transient tables

Other tables will be effectively hidden! Relevant for time travel.

Not visible to other users!

- ✓ Not possible to change type of object for existing object





Zero-Copy Cloning

```
CREATE TABLE table_new  
CLONE table_source
```

New table name

Source for clone

Cloning a database or schema will
clone all contained objects

Cloning Syntax

Database

Schema

Table

Stream

File Format

Sequence

Stage

Task

Pipe

Named internal stages
not cloned

Only for external stages





Zero-Copy Cloning

Domain 6.0:
Data Protection and Data Sharing



01

02

03

04

05

06





Zero-Copy Cloning



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Zero-Copy Cloning

- ✓ Create copies of a database, a schema or a table



003-1040559

1250 003-77156.8

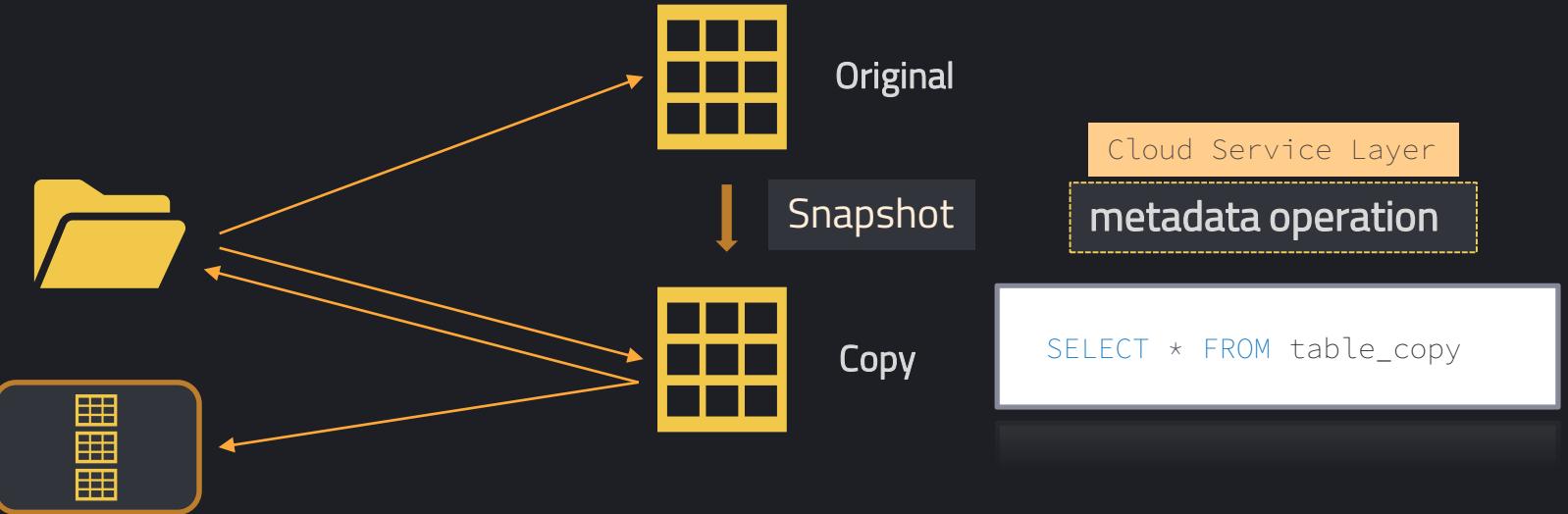
1760 0009-14563.7 73273





Zero-Copy Cloning

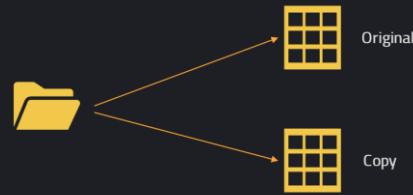
- ✓ Create copies of a database, a schema or a table





Zero-Copy Cloning

- ✓ Create copies of a database, a schema or a table

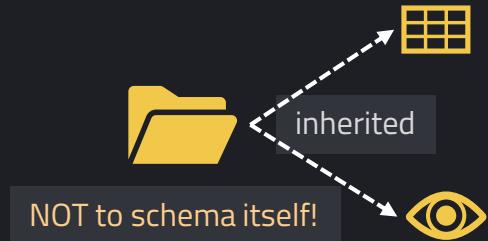
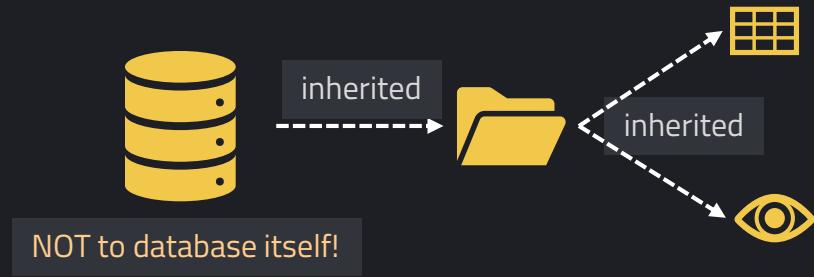


- ✓ Cloned object is independent from original table
- ✓ Easy to copy all metadata & improved storage management
- ✓ Creating backups for development purposes
- ✓ Typically combined with time travel



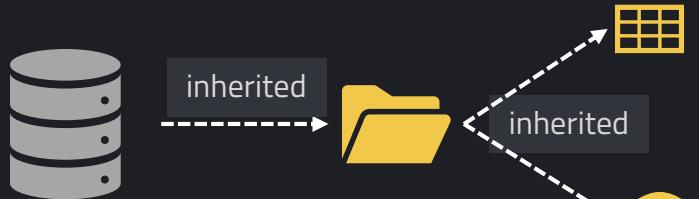


How about privileges?



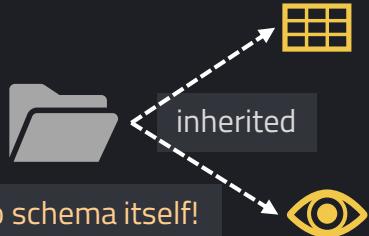


How about privileges?



NOT to database itself!

Privileges will always only be inherited to child objects never to source object itself



NOT to schema itself!



What privileges are needed?

Table

SELECT

Pipe

Stream

Task

OWNER

All other
objects

USAGE





Additional Considerations

- ✓ Load history metadata is not copied

Loaded data can be loaded again

```
CREATE TABLE table_new  
CLONE table_source  
BEFORE (TIMESTAMP => 'timestamp')
```

Cloning from specific point in time is possible.





Zero-Copy Cloning

```
CREATE TABLE table_new  
CLONE table_source  
BEFORE (TIMESTAMP => 'timestamp')
```

Cloning from specific point in time is possible.





Data Sharing



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Data Sharing

Usually this can be also a rather complicated process....

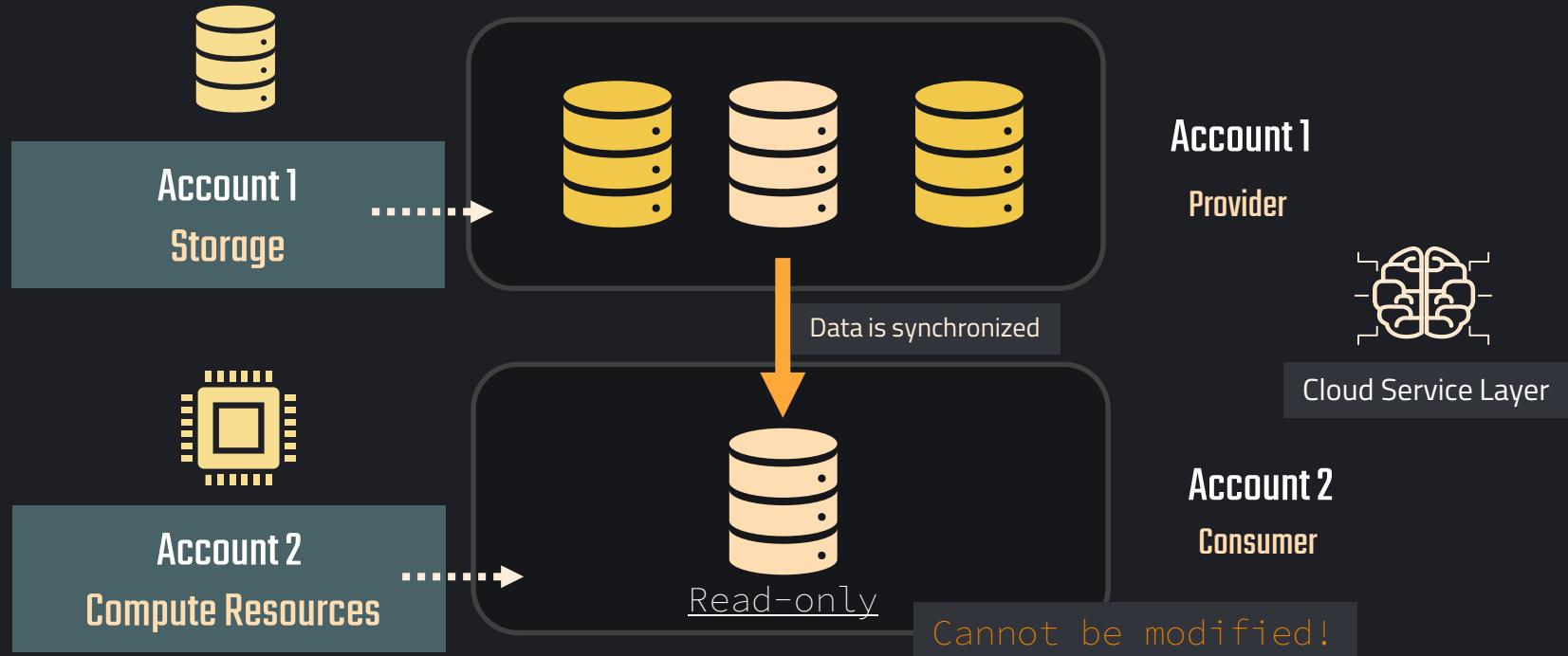
- ✓ Sharing with actually copying data
- ✓ Data is always up-to-date
- ✓ Compute paid by consumer





Data Sharing

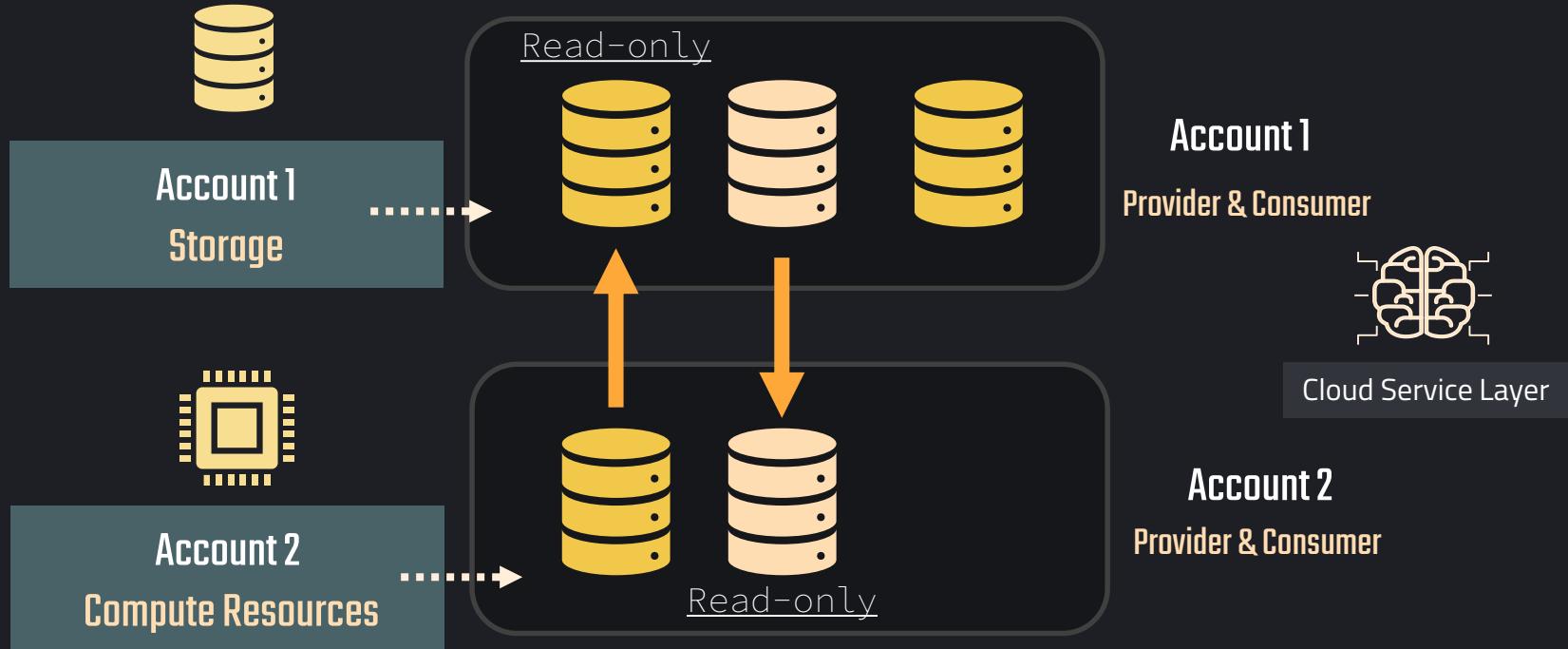
Standard Edition





Data Sharing

Standard Edition





Setting up share

1. Create share

ACCOUNTADMIN role or CREATE SHARE privileges required

```
CREATE SHARE my_share;
```

2. Grant privileges to share

```
GRANT USAGE ON DATABASE my_db TO SHARE my_share;  
GRANT USAGE ON SCHEMA my_schema.my_db TO SHARE my_share;  
GRANT SELECT ON TABLE my_table.myschema.my_db TO SHARE my_share;
```

3. Add consumer account(s)

```
ALTER SHARE my_share ADD ACCOUNT bl67131;
```

4. Import share

ACCOUNTADMIN role or IMPORT SHARE / CREATE DATABASE privileges required

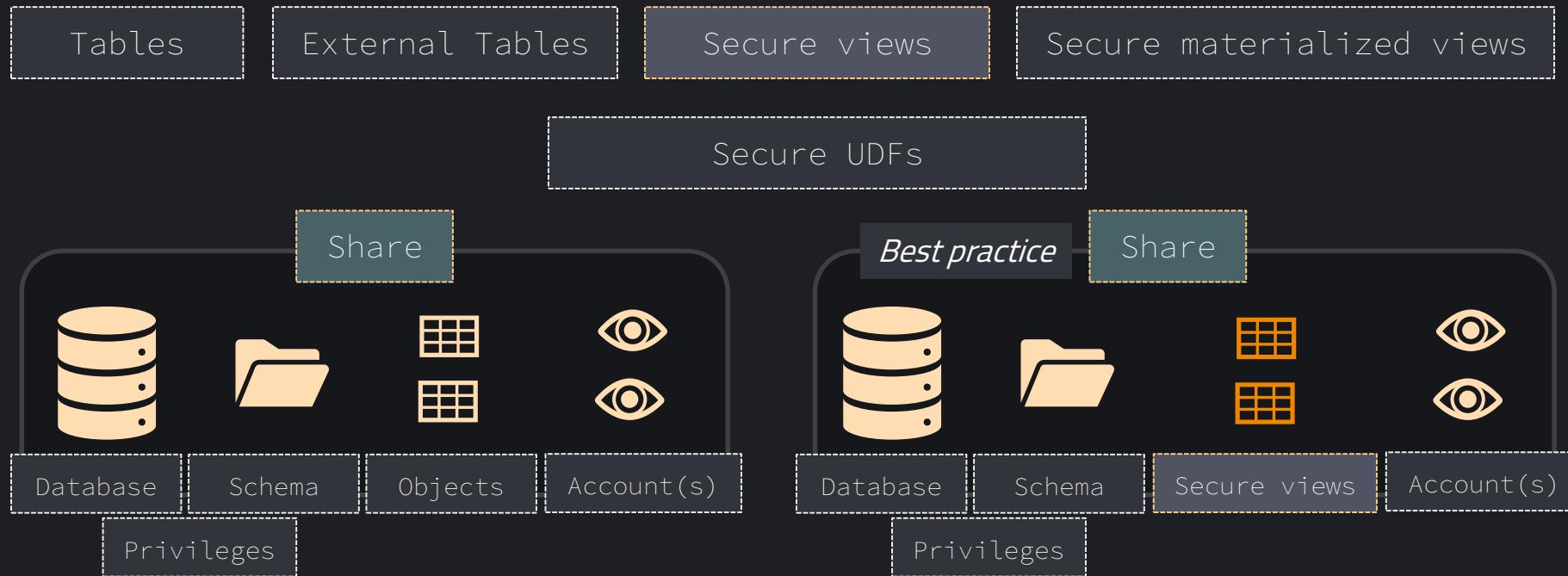
```
CREATE DATABASE my_db FROM SHARE my_share;
```

5. Grant privileges

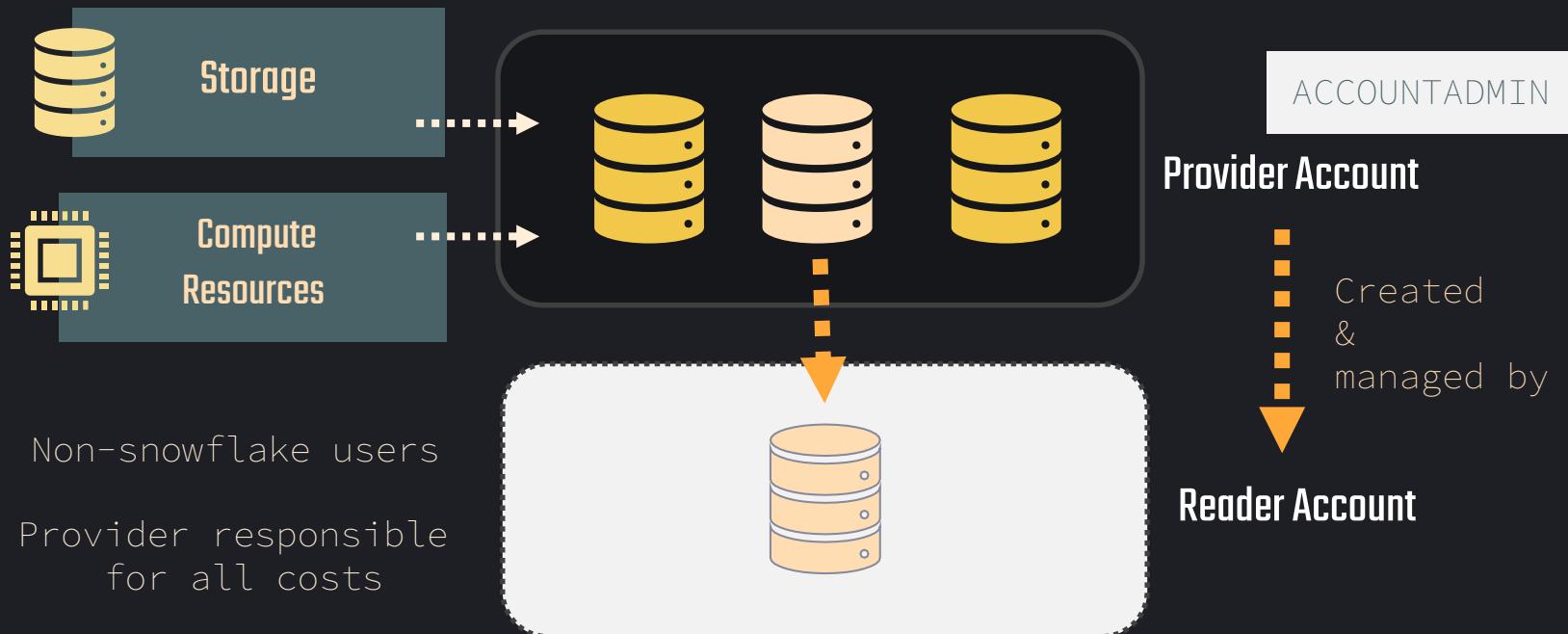




What can be shared

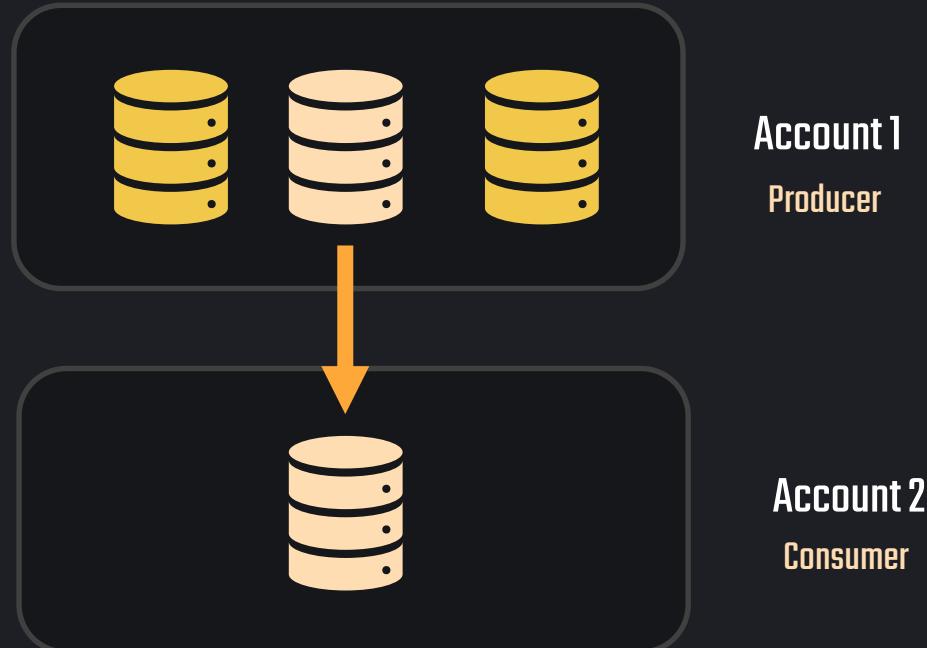


Data Sharing with Non-Snowflake Users



Data Sharing Considerations

- ✓ Share becomes immediately visible once shared
New objects added immediately visible as well
- ✓ Each account can share and consume
Even own share can be consumed
- ✓ Virtual Private Edition doesn't allow sharing
Dedicated compute and metadata storage
- ✓ Marketplace: Find & Import third-party datasets
ACCOUNTADMIN role or IMPORT SHARE privileges required
- ✓ Data Exchange: Private Hub for sharing data
Members can be invited
Needs to be enabled by reaching out to Snowflake support





Data Sharing Considerations

- ✓ Share becomes immediately visible once shared
 - New objects added immediately visible as well
- ✓ Each account can share and consume
 - Even own share can be consumed
- ✓ Sharing across region / cloud provider must be enabled
 - Done via replicating
- ✓ Virtual Private Edition doesn't allow sharing
 - Dedicated compute and metadata storage
- ✓ Marketplace: Find & Import third-party datasets
 - ACCOUNTADMIN role or IMPORT SHARE privileges required
- ✓ Data Exchange: Private Hub for sharing data
 - Members can be invited
 - Needs to be enabled by reaching out to Snowflake support





Sharing with Non Snowflake users

New Reader
Account

- ✓ Independant instance with own url & own compute resources

Share data

- ✓ Share database & table

Create Users

- ✓ As administrator create user & roles

Create database

- ✓ In reader account create database from share





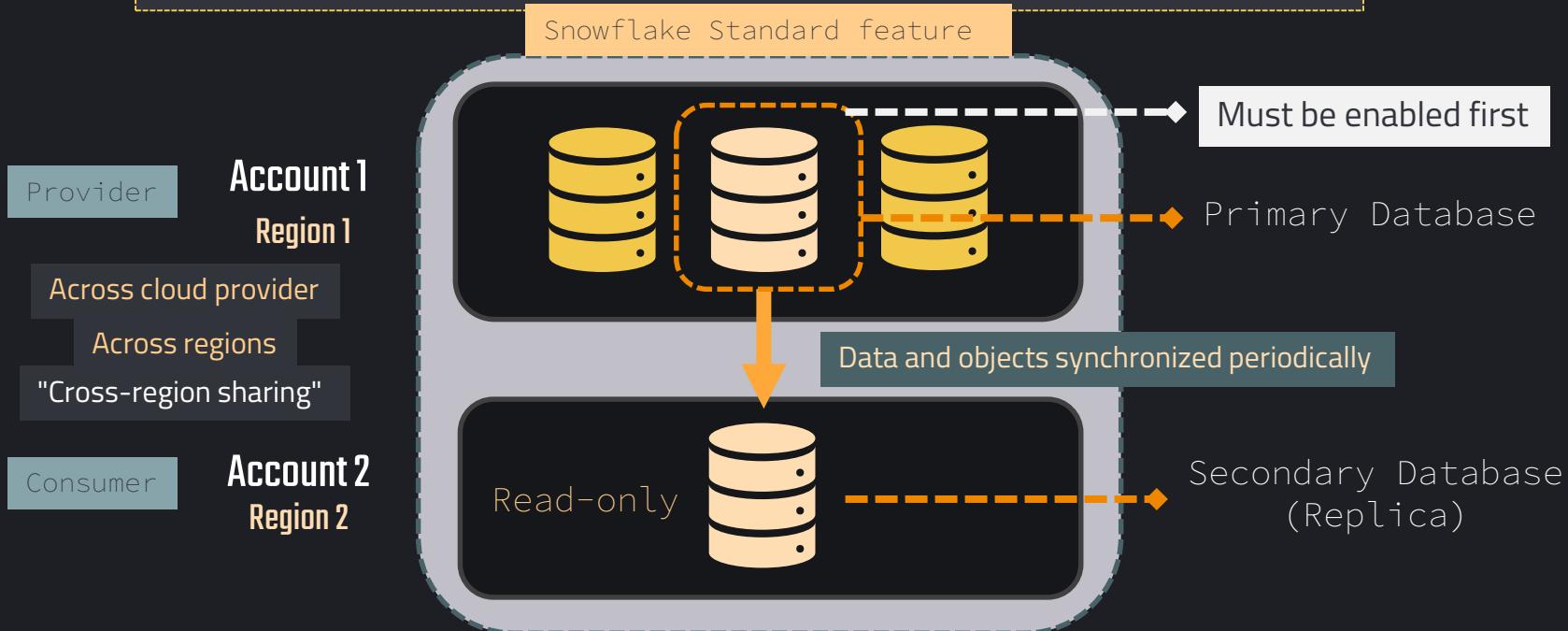
Database replication





Database replication

Replicates a database between accounts within the same organization.





Database replication

1. Enable replication for source account with ORGADMIN role

```
show organization accounts;

-- Enable replication for each source and target account in your organization
select system$global_account_set_parameter('<organization_name>.<account_name>',
                                           'ENABLE_ACCOUNT_DATABASE_REPLICATION', 'true');
```

Step 2: Promote a Local Database to Primary Database with ACCOUNTADMIN role

```
ALTER DATABASE my_db1 ENABLE REPLICATION TO ACCOUNTS myorg.account2, myorg.account3;
```

Step 3: Create replica in consumer account

```
CREATE DATABASE my_db1 AS REPLICA OF myorg.account1.my_db1;
```





Database replication

Step 4: Refresh database

```
ALTER DATABASE my_db1 REFRESH;
```

Ownership privileges are needed

A task can be scheduled with this command





Database replication

✓ Privileges must be granted separately (not inherited)

✓ All objects in database are replicated apart from:

Temporary tables

Stages

Pipes

✓ Data Transfer cost apply according to cloud provider

✓ Compute cost apply

✓ Data is actually physically replicated





Account & Security

Domain 2.0:
Account Access and Security



01

02

03

04

05

06

06



Acess Control



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Access Control

Two aspects

Discretionary Access Control (DAC)

Each object has an owner

Owner can grant access to that object

Role-based Access Control (RBAC)

Privileges

Roles

Users





Access Control

Key Concepts



Securable object

Access can be granted

Access denied unless granted



Privilege

Defined level of access



Role

Entity to which privileges are granted

Will be assigned to users ... or other roles

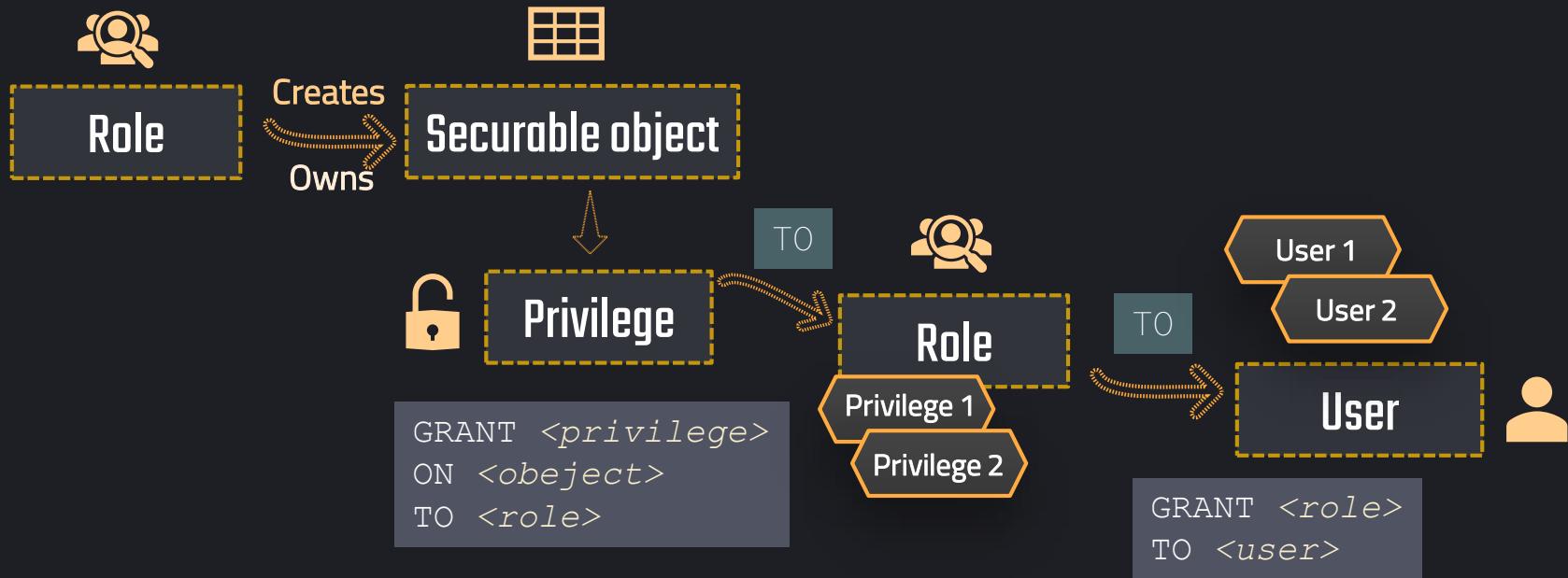


User

Identity associated with person or program



Access Control



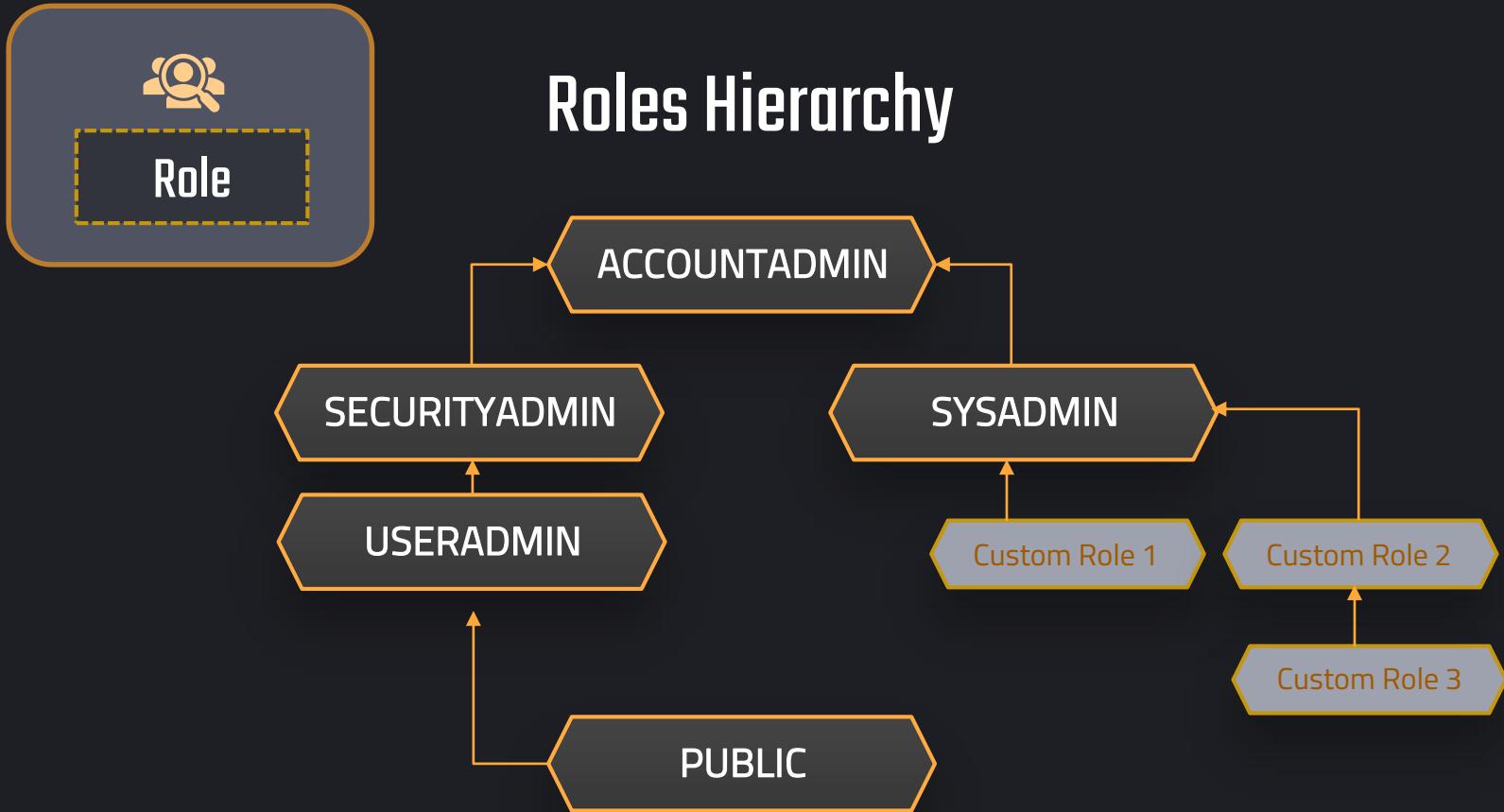
003-1040559

1250 003-77156.8

1760 0009-14563.7

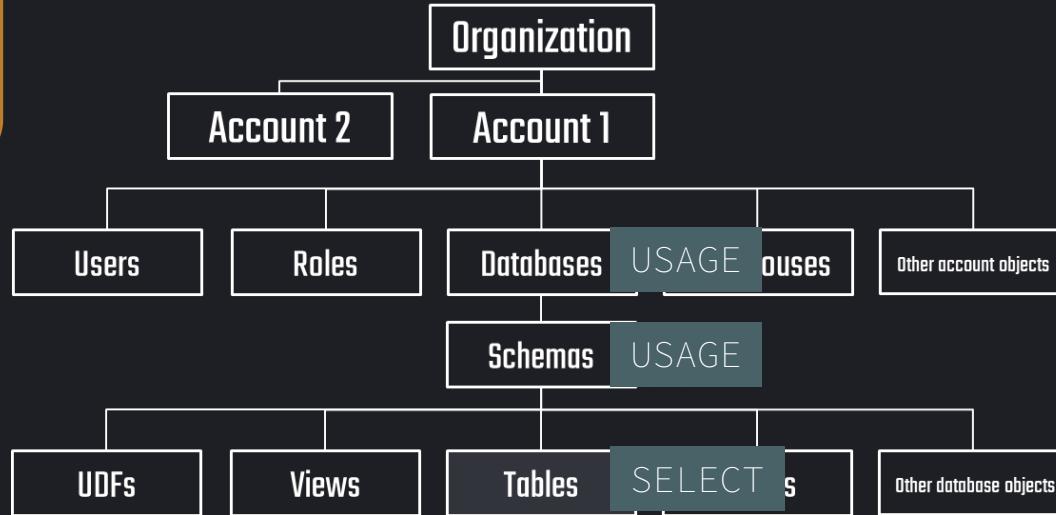
73273







Objects in Snowflake





Objects in Snowflake



✓ Every object is owned by one single role

- All privileges per default
- Including GRANT and REVOKE
- Active role in the session

OWNERSHIP privileges

✓ Ownership can be transferred





Roles



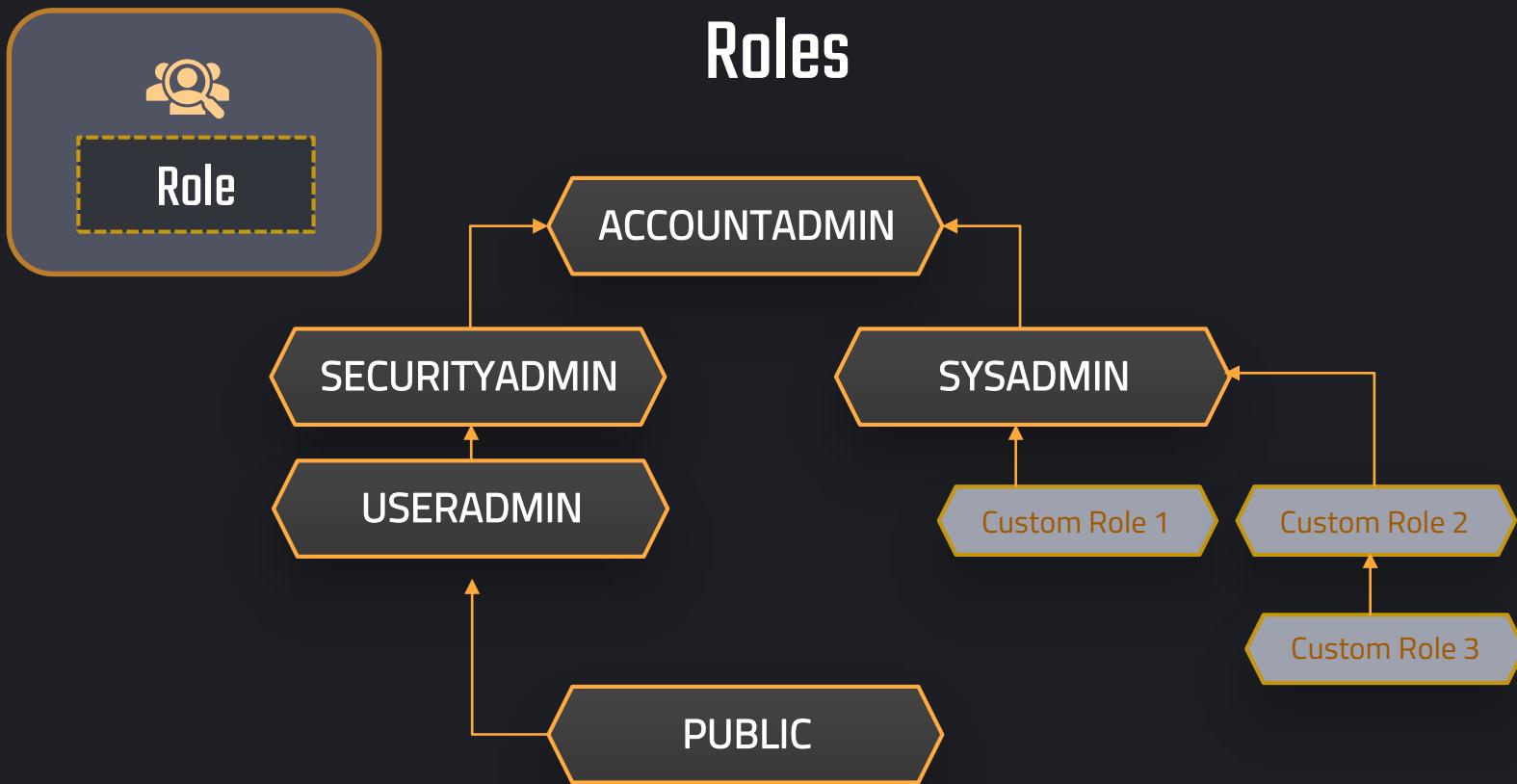
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

Roles





Roles



GRANT
REVOKE

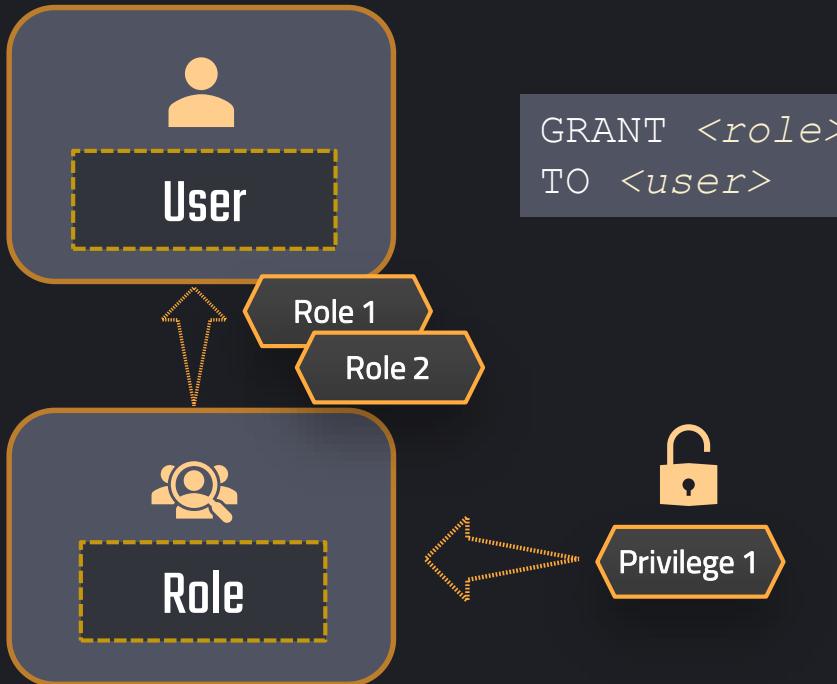
```
GRANT <privilege>  
ON <object>  
TO <role>
```

SELECT
my_table





Roles



- ✓ Roles are assigned to users
- ✓ Multiple roles can be assigned





Roles

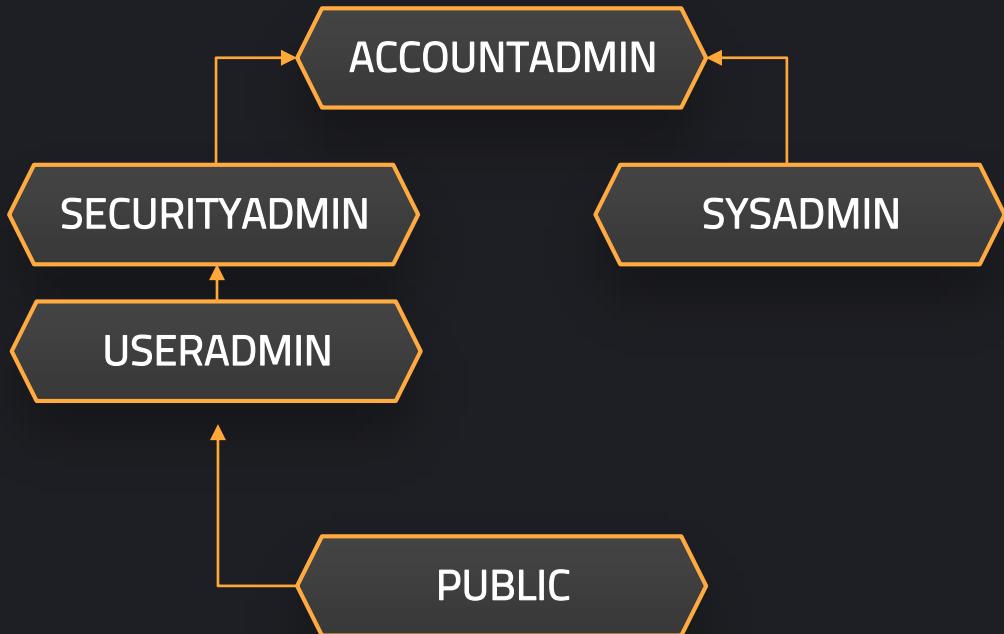


✓ "Current role" in every session
⇒ Primary role





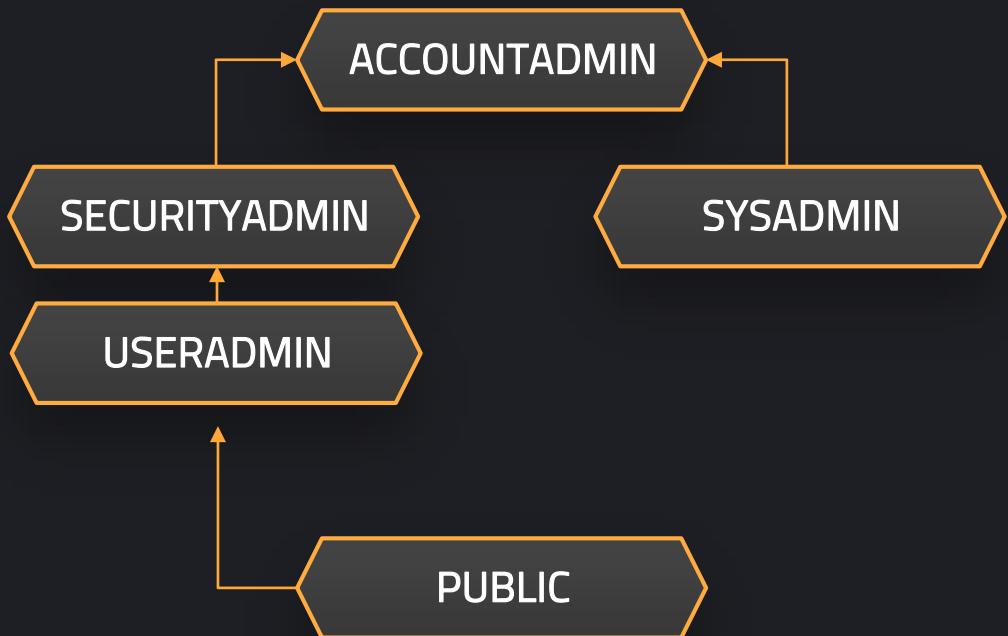
System-defined Roles



- ✓ Can't be dropped
- ✓ Privileges can be added but not revoked



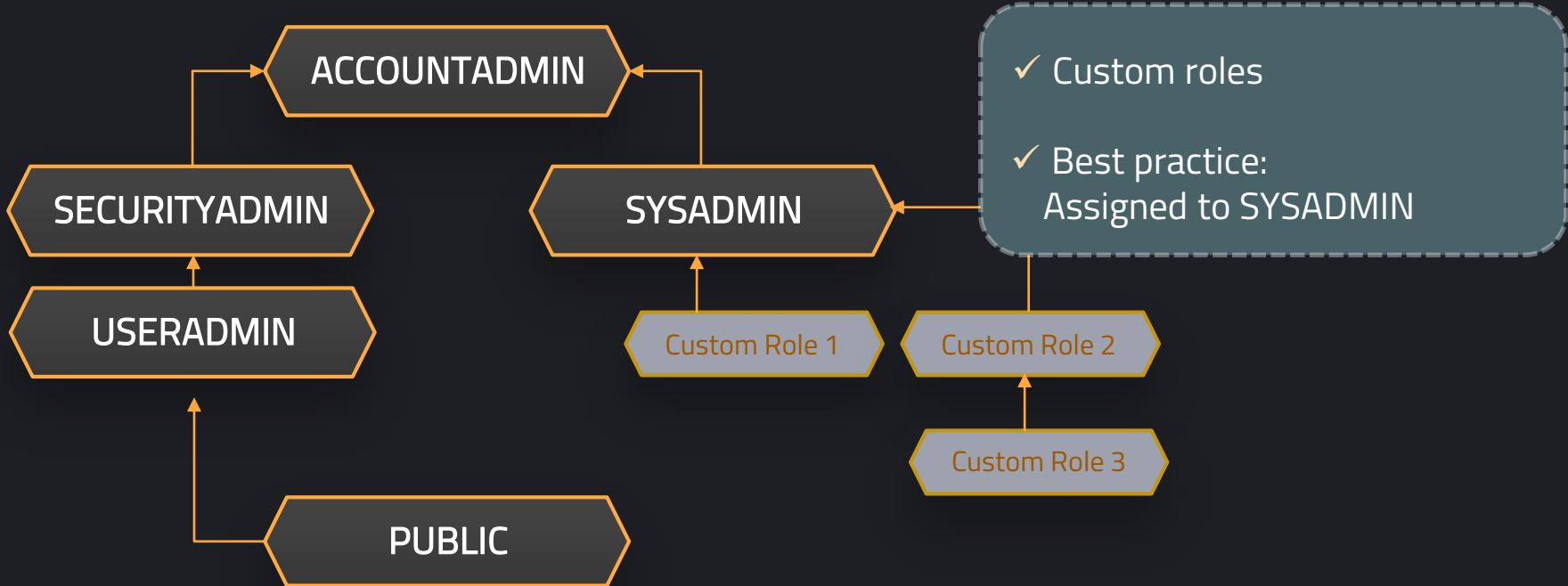
Roles



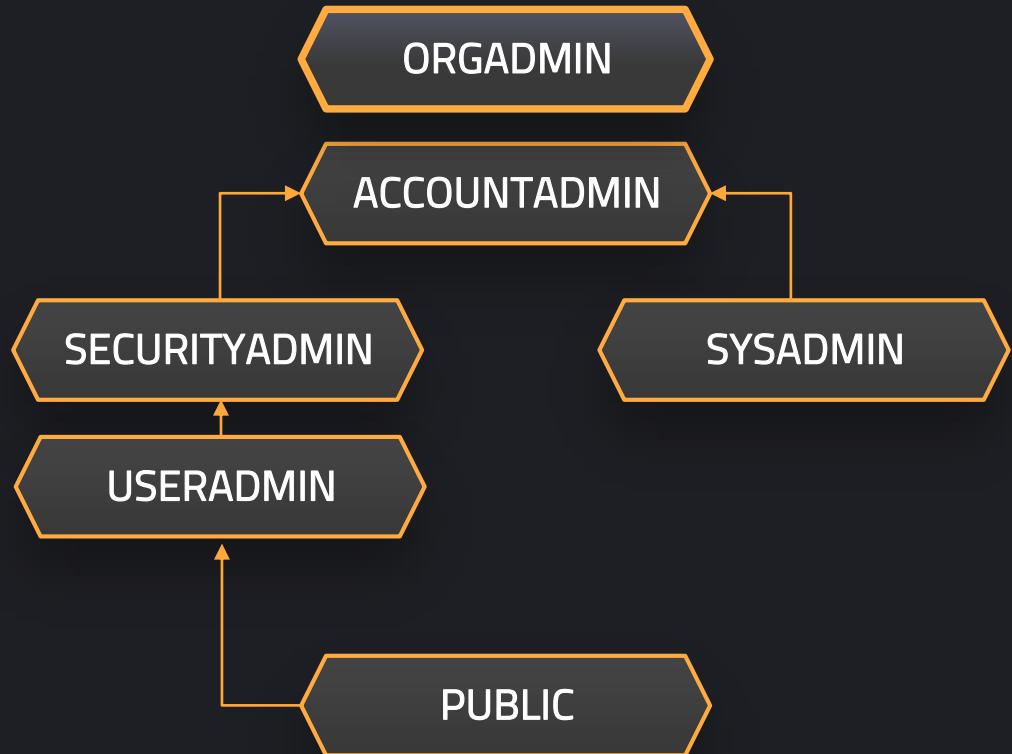
- ✓ Roles can be assigned to roles
- ✓ Hierarchy of roles
- ✓ Privileges are inherited



Roles



Roles



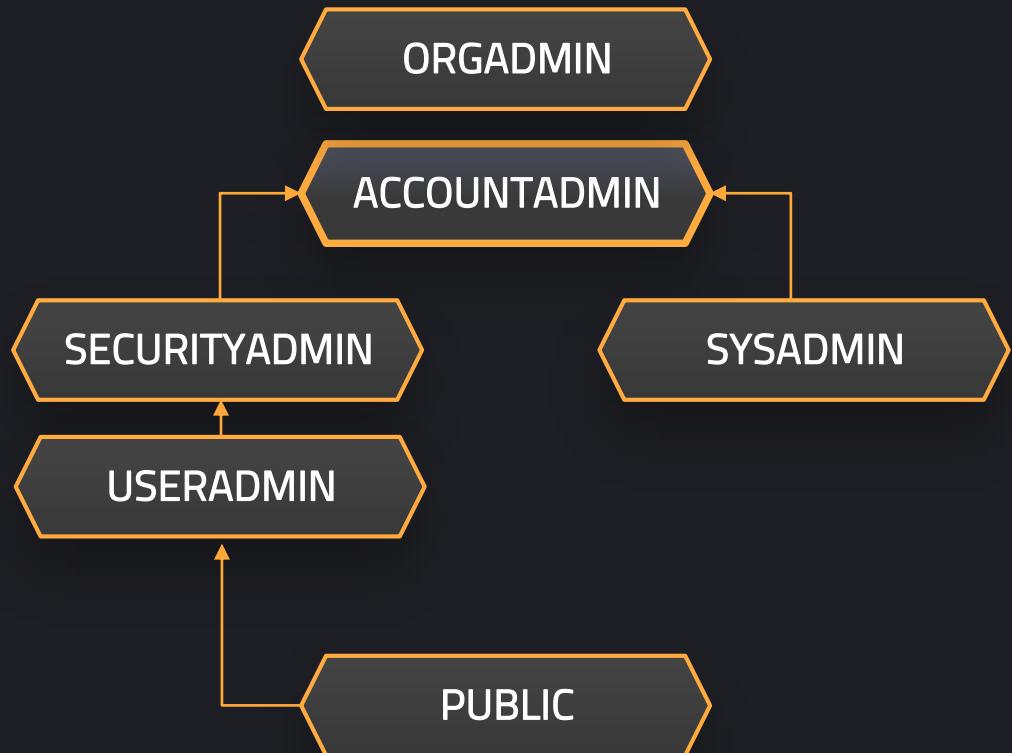
ORGADMIN

Manages actions on organizational level.

- ✓ Create accounts
- ✓ View all accounts
- ✓ View account usage information



Roles

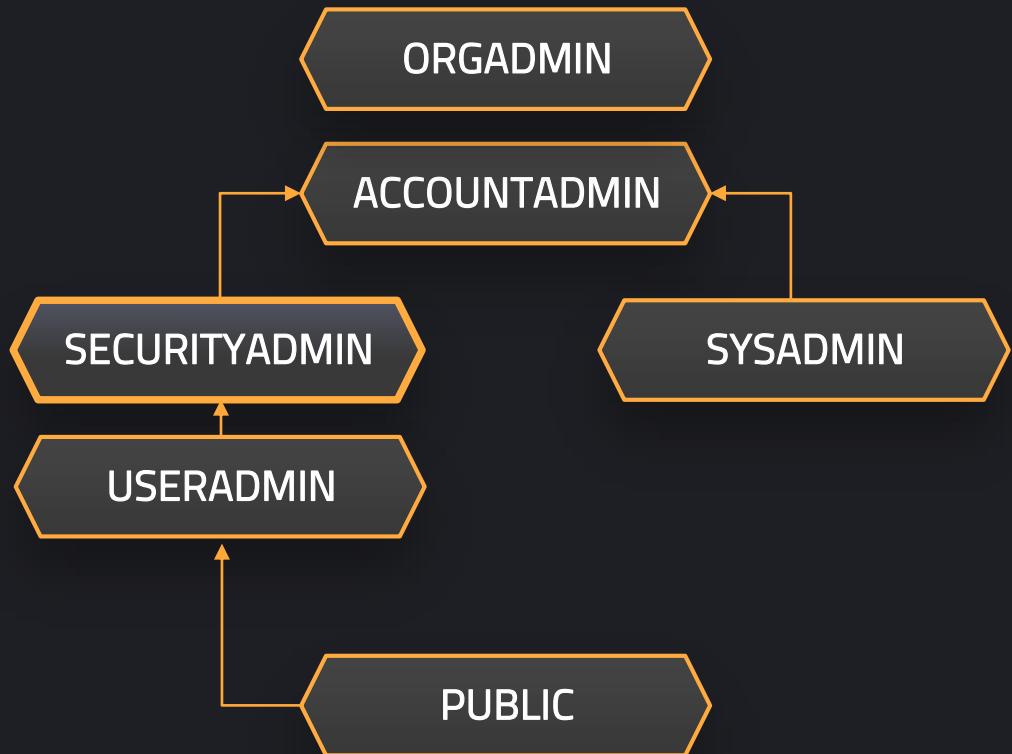


ACCOUNTADMIN

Top-level role

- ✓ Should be to limited number of users
- ✓ Contains SECURITYADMIN & SYSADMIN
- ✓ Can manage all objects in account
- ✓ Incl. share and reader accounts
- ✓ Modify account-level parameters
- ✓ Manage billing & resource monitors

Roles

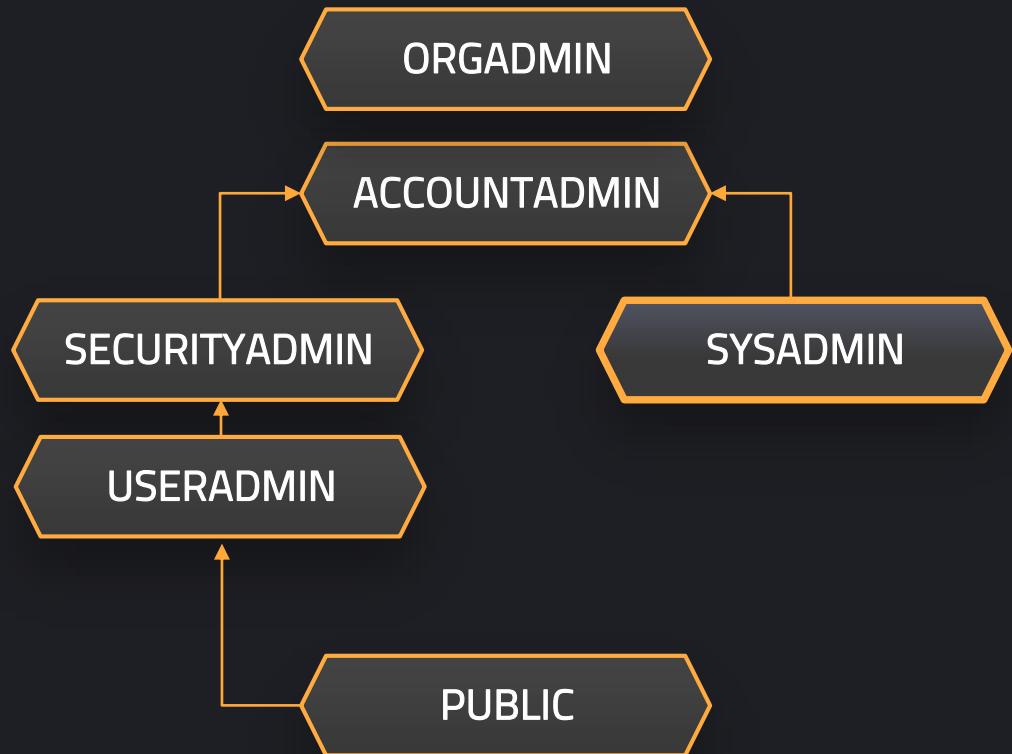


SECURITYADMIN

Manage any object grant globally

- ✓ MANAGE GRANTS privilege
- ✓ Create, monitor, and manage users & roles
- ✓ Inherits USERADMIN privileges

Roles

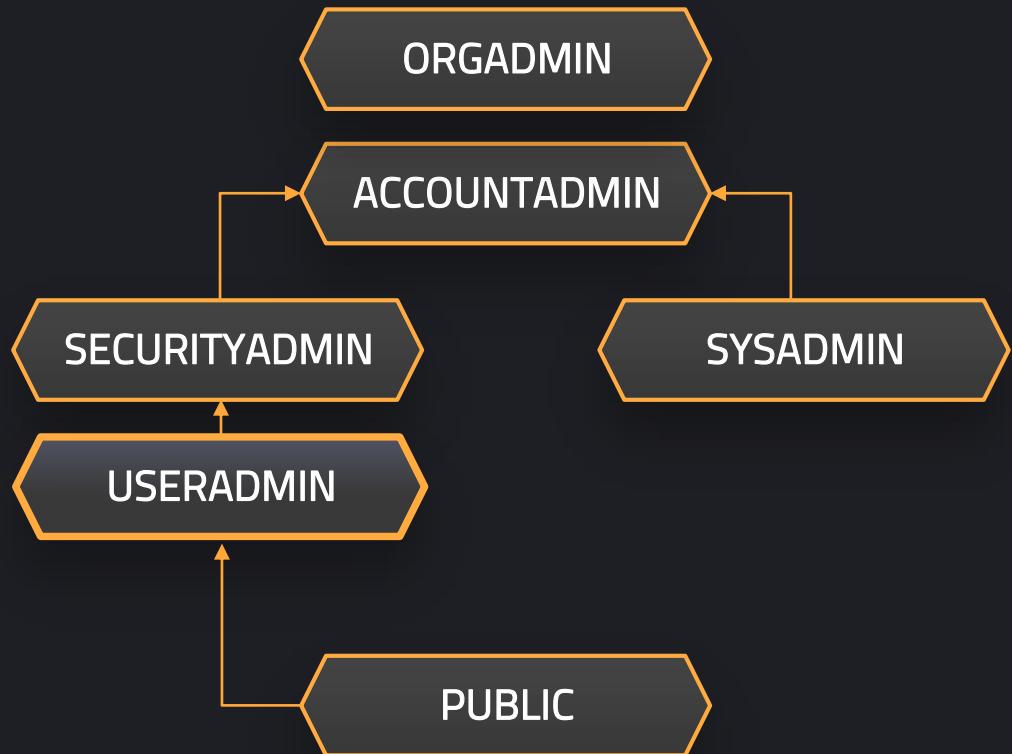


SYSADMIN

Create warehouses, databases & other objects

- ✓ All custom roles should be assigned to
- ✓ Can grant privileges on warehouses, databases, and other objects

Roles

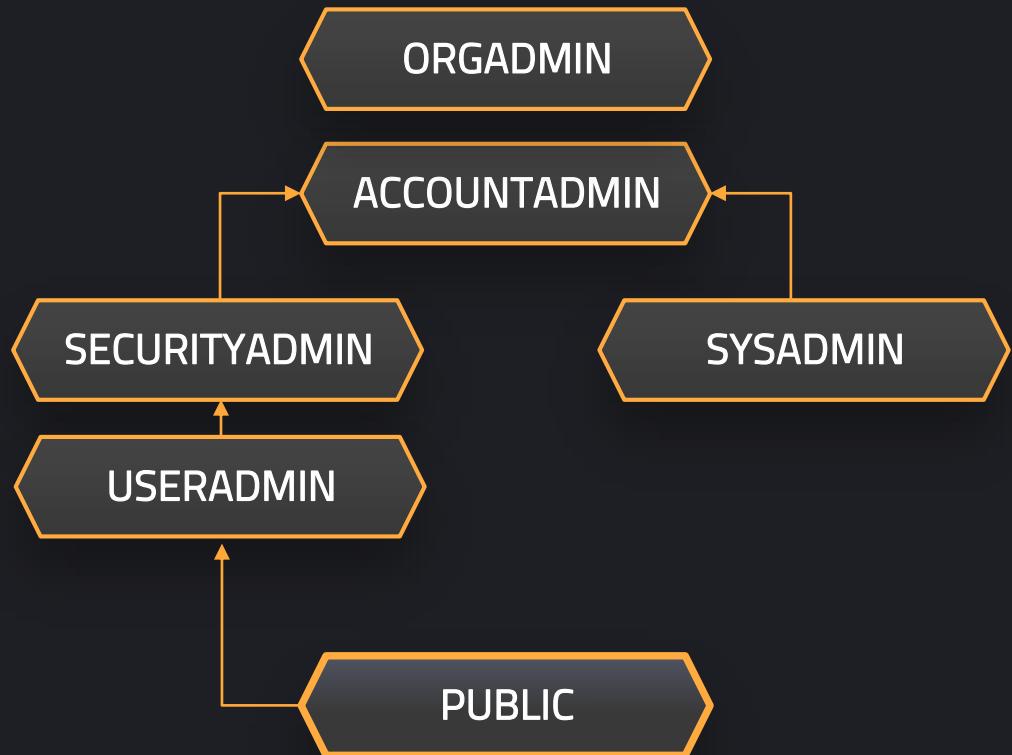


USERADMIN

Dedicated to user and role management

- ✓ CREATE USER & CREATE ROLE privileges
- ✓ Can manage users and roles that are owned

Roles

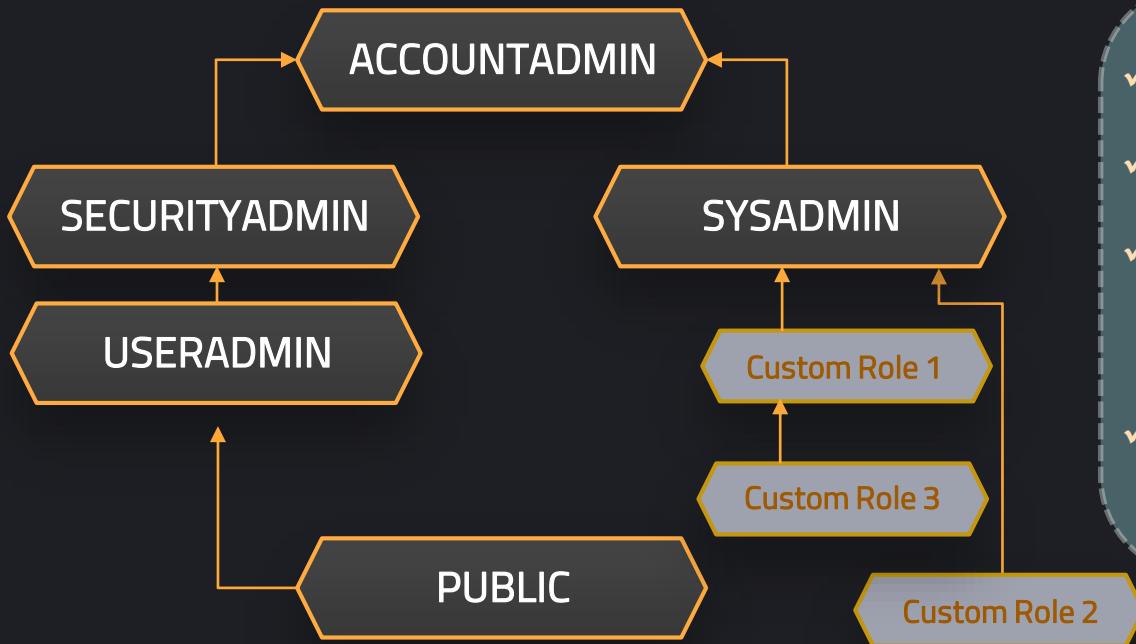


PUBLIC

Automatically granted per default

- ✓ Granted to when no access control needed
- ✓ Objects can be owned but are available to everyone

Roles





Privileges



003-1040559

1250 003-77156.8

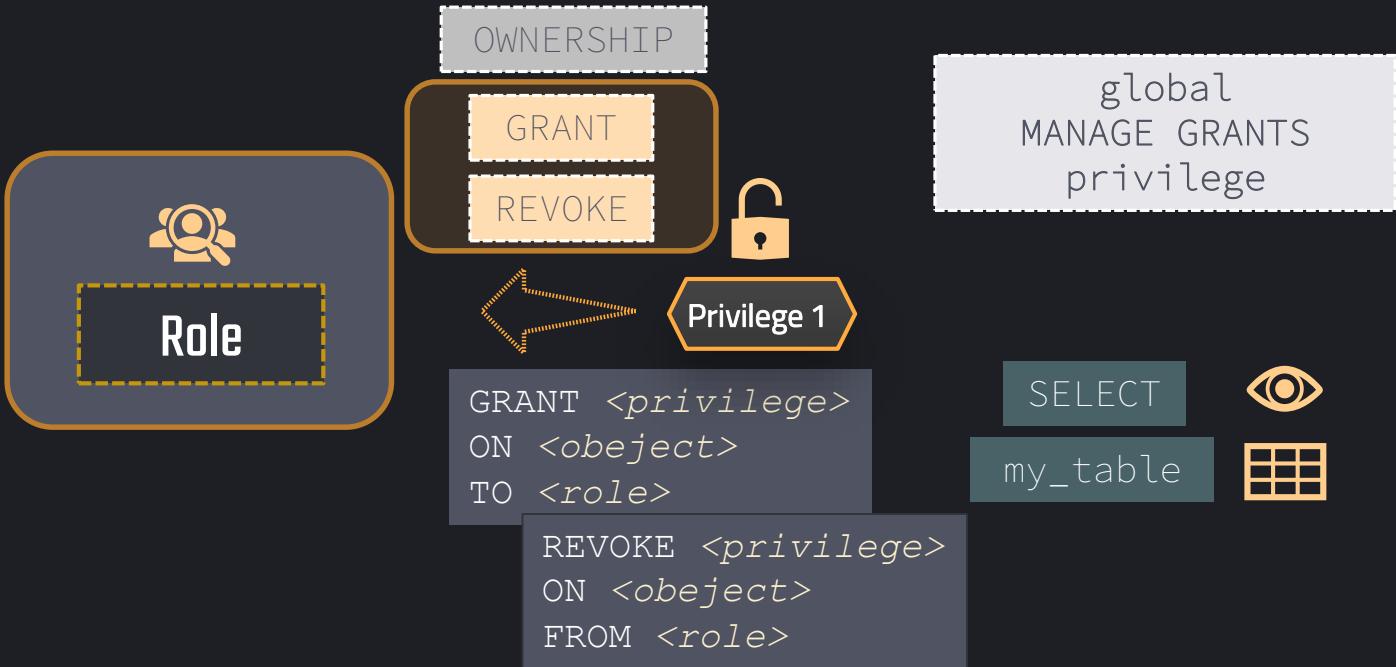
1760 0009-14563.7

73273



Privileges

Define granular level of access





Privileges

Important privileges

Global privileges

CREATE SHARE

IMPORT SHARE

APPLY MASKING POLICY

Enables provider to create a share.

Enables to create a database.

Enables to set masking policies.





Privileges

Important privileges

Virtual Warehouse

MODIFY

Enables to alter properties of a warehouse – e.g. resizing.

MONITOR

Enables to view executed queries by the warehouse.

OPERATE

Enables to change the state of a warehouse (e.g. suspend and resume).

USAGE

Enables to use the warehouse and execute queries.

OWNERSHIP

Full control over the warehouse.

ALL

All privileges apart from OWNERSHIP.





Privileges

Important privileges

Databases

MODIFY

Enables to alter properties and settings of a database.

MONITOR

Enables to perform DESCRIBE command.

USAGE

Enables to use the database and execute SHOW DATABASES command.

REFERENCE_USAGE

Enables using an object (shared secure view) to reference another object in a different database.

ALL

All privileges apart from OWNERSHIP.

OWNERSHIP

Full control over the database.

CREATE SCHEMA

Enable creating a schema in the database.





Privileges

Important privileges

Stages

READ

Enables to perform operations that require reading (e.g. GET, LIST, COPY INTO table) from internal stages; not applicable to external stages

USAGE

Enables to use an external stage; not applicable to internal stage.

WRITE

Enables to perform writing to internal stage (PUT, REMOVE, COPY INTO location); not applicable to external stages

ALL

All privileges apart from OWNERSHIP.

OWNERSHIP

Full control over the stage.





Privileges

Important privileges

Tables

SELECT

Using SELECT to query table.

INSERT

Inserting values into the table and manually reclustering tables.

UPDATE

Using UPDATE command on a table.

TRUNCATE

Using TRUNCATE command on a table.

DELETE

Using DELETE command on a table.

ALL

All privileges apart from OWNERSHIP.

OWNERSHIP

Full control over the database.





Multi-factor authentication

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Multi-Factor Authentication

Authentication is proving that you are who you say you are.

Multi-Factor Authentication provides additional login security.

Standard Edition

Powered by Duo Security but managed by Snowflake.

No sign-up,
only installation.



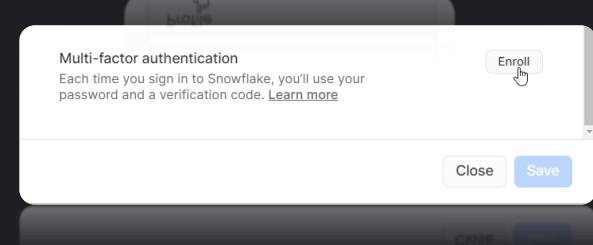
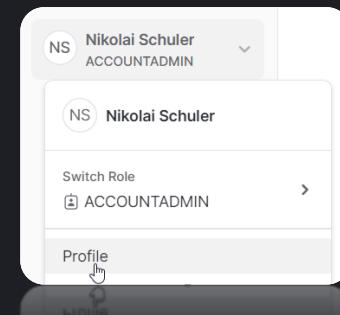
Multi-Factor Authentication

Per default enabled for accounts but requires user to [enroll](#).

SECURITYADMIN (or ACCOUNTADMIN) can disable MFA for user.

Fully-supported by web interface, SnowSQL, Snowflake ODBC and JDBC, and Python Connectors.

Strongly recommended for
ACCOUNTADMIN





Multi-Factor Authentication

MFA **token caching** can reduce the number of prompts during authentication.

Needs to be enabled first.

MFA token is valid for up to 4 hours.

ODBC driver version 2.23.0 (or later).

JDBC driver version 3.12.16 (or later).

Python Connector for Snowflake version 2.3.7 (or later).





Federated Authentication (SSO)



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Federated Authentication (SSO)

Enables users to use login via SSO.

- ✓ Login
- ✓ Logout
- ✓ Time out due to inactivity

Service provider
(Snowflake)

External Identity provider
(IdP)

Federated
environment

Maintaining credentials

Authenticate users

- ✓ most **SAML 2.0-compliant** vendors are supported as an IdP are supported
- ✓ Native support for **Okta** and **Microsoft AD FS**





SSO-Login Workflow

Snowflake-initiated

User navigates to Snowflake WebUI.

Choose login via configured IdP (e.g. Okta or AD FS)

Authenticate via IdP credentials (e.g. email and password)

External Identity provider
(IdP)



Snowflake

Opens Snowflake session





SSO-Login Workflow

IdP-initiated

User navigates to IdP

Authenticate with credentials

Select Snowflake as application

External Identity provider
(IdP)



Snowflake

Opens Snowflake session





SCIM support

Snowflake is compatible with SCIM 2.0

SCIM is an open standard for automating user provisioning.

Create user in IdP



Provision
User

Snowflake





Key Pair Authentication



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Key Pair Authentication

Enhanced security as an alternative to basic username/password.

One or two



Public Key

Public Key

Private Key

Minimum:
2048-bit RSA
key pair

Connecting via Snowflake
Clients (SnowSQL etc.)





Key Pair Authentication

Enhanced security as an alternative to basic username/password.

1. Generate Private Key

2. Generate Public Key

3. Store the Keys locally

4. Assign public key to user

5. Configure client to use
key pair authentication

```
ALTER USER my_user SET  
RSA_PUBLIC_KEY 'FGKdojfeFdD...';
```





Column-level security



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Column-level Security

Column-level security masks data in tables and views enforced on columns

Enterprise Edition

Dynamic Data Masking

Masking policy

Based on

Role

Masked at runtime

PHONE
##-###-##
##-###-##
##-###-##

FULL_NAME	EMAIL	PHONE
Lewiss MacDwyer	lmacdwyer0@un.org	262-665-9168
Ty Pettingall	tpettingall1@mayoclinic.com	734-987-7120
Marlee Spadazzi	mspadazzi2@txnews.com	867-946-3659
M.....	m.....@m.....com	801-840-3820

FULL_NAME	EMAIL	PHONE
L*****	l*****	##-##-##
T*****	t*****	##-##-##
M*****	m*****	##-##-##
M.....	m.....	##-##-##

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Column-level Security

Column-level security masks data in tables and views enforced on columns

Dynamic Data Masking

Original column value

```
CREATE MASKING POLICY my_policy  
AS (val varchar) returns varchar ->  
CASE  
WHEN current_role in (role_name)  
THEN val  
ELSE '##-##'  
END;
```

Define policy

Masking value

Apply policy

```
ALTER TABLE my_table MODIFY COLUMN phone  
SET MASKING POLICY my_policy;
```





Column-level Security

Column-level security masks data in tables and views enforced on columns

Dynamic Data Masking

Original column value

```
CREATE MASKING POLICY my_policy  
AS (val varchar) returns varchar ->  
CASE  
WHEN current_role in (role_name)  
THEN val  
ELSE '##-##'  
END;
```

Define policy

Masking value

Apply policy

```
ALTER TABLE my_table MODIFY COLUMN phone  
UNSET MASKING POLICY my_policy;
```





Column-level Security

Column-level security masks data in tables and views enforced on columns

Enterprise Edition

External Tokenization

Data is tokenized

- ✓ Analytical value preserved
- ✓ Sensitive information protected

Pre-load

- ✓ Data is tokenized pre-load and detokenized at query runtime





Row-level security



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Row-Level Security

Supported through row access policies to determine which rows are returned

Enterprise Edition

Row Access Policies

Filtered at runtime

Condition

User

Role

	ORDER_ID	AMOUNT	PROFIT	QUANTITY	CATEGORY	SUBCATEGORY
1	B-25601	1,275	-1,148	7	Furniture	Bookcases
2	B-25601	66	-12	5	Clothing	Stole
3	B-25601	8	-2	3	Clothing	Hankerchief
4	B-25601	80	-56	4	Electronics	Electronic Games
	B-52001	80	-20	4	Electronics	Electronic Games

	ORDER_ID	AMOUNT	PROFIT	QUANTITY	CATEGORY	...	SUBCATEGORY
	B-25601	1,275	-1,148	7	Furniture		Bookcases
	B-25603	24	-30	1	Furniture		Chairs
	B-25608	1,364	864	5	Furniture		Tables
	B-25608	476	0	3	Furniture		Chairs
	B-52008	418	0	3	Furniture		Chairs
	003-1040559	1250	003-77156.8	1760	0009-14563.7		73273





Row-Level Security

Define policy

Signature column

```
CREATE ROW ACCESS POLICY my_policy
AS (column1 varchar) returns boolean ->
CASE
WHEN !ROLE_NAME' = current_role()
    and 'value1'=column1 THEN true
ELSE false
END;
```



	ORDER_ID	AMOUNT	PROFIT	QUANTITY	CATEGORY	SUBCATEGORY
1	B-25601	1,275	-1,148	7	Furniture	Bookcases
2	B-25601	66	-12	5	Clothing	Stole
3	B-25601	8	-2	3	Clothing	Hankerchief
3	B-25601	8	-3	3	Clothing	Hankerchief

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273



Row-Level Security

Signature column

Define policy

```
CREATE ROW ACCESS POLICY my_policy  
AS (column1 varchar) returns boolean ->  
CASE  
WHEN !ROLE_NAME' = current_role()  
and 'value1'=column1 THEN true  
ELSE false  
END;
```

Apply policy

```
ALTER TABLE my_table ADD ROW POLICY my_policy  
ON (column1);
```





Row-Level Security

Signature column

Define policy

```
CREATE ROW ACCESS POLICY my_policy  
AS (column1 varchar) returns boolean ->  
CASE  
WHEN !ROLE_NAME' = current_role()  
and 'value1'=column1 THEN true  
ELSE false  
END;
```

Apply policy

```
ALTER TABLE my_table DROP ROW POLICY my_policy;
```





Network policies



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Network policies

Allow to restrict access to account based on user IP addresses

Standard Edition

Allowed IP addddresses

IP_address1, IP_address2, IP_address3,

Blocked IP addresses

IP_address2

Has priority





Network policies

Allow to restrict access to account based on user IP addresses

Standard Edition

Allowed IP adddresses

192.168.1.0/24

(192.168.1.0 - 192.168.1.255)

Blocked IP addresses

192.168.1.75, 192.168.1.15

Has priority

CIDR notation





Network policies

Allow to restrict access to account based on user IP addresses

Standard Edition

Allowed IP adddressses

192.168.1.95, 192.168.1.113

Blocked IP addresses

N/A

Has priority





Network policies

Create Network Policy

SECURITYADMIN

global CREATE NETWORK POLICY privilege

```
CREATE NETWORK POLICY my_network_policy  
ALLOWED_IP_LIST = ('192.168.1.95', '192.168.1.113');
```





Network policies

Create Network Policy

SECURITYADMIN

global CREATE NETWORK POLICY privilege

```
CREATE NETWORK POLICY my_network_policy  
ALLOWED_IP_LIST = ('192.168.1.95', '192.168.1.113'),  
BLOCKED_IP_LIST = ('192.168.1.95');
```





Network policies

Apply Network Policy

SECURITYADMIN

global CREATE NETWORK POLICY privilege

```
ALTER ACCOUNT SET NETWORK_POLICY = mynetwork_policy;
```





Network policies

Apply Network Policy

SECURITYADMIN

global CREATE NETWORK POLICY privilege

```
ALTER ACCOUNT UNSET NETWORK_POLICY;
```





Network policies

Apply Network Policy to USER

OWNERSHIP of User & Network Policy

```
ALTER USER SET NETWORK_POLICY = mynetwork_policy;
```





Data Encryption

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Data Encryption

All data is encrypted at rest and in transit

Standard Edition

Encryption at rest



Automatically by default

Tables

Internal Stages



AES 256-bit encryption

Enterprise Edition

If enabled

Re-keying every year

Key Rotation



Snowflake-managed

Key Rotation every 30 days

Old keys will be destroyed





Data Encryption

All data is encrypted at rest and in transit

Automatically by default

Standard Edition

Data in Transit

WebUI

SnowSQL

JDBC

ODBC

Python Connector



TLS 1.2

End-to-End encryption



003-1040559

1250 003-77156.8

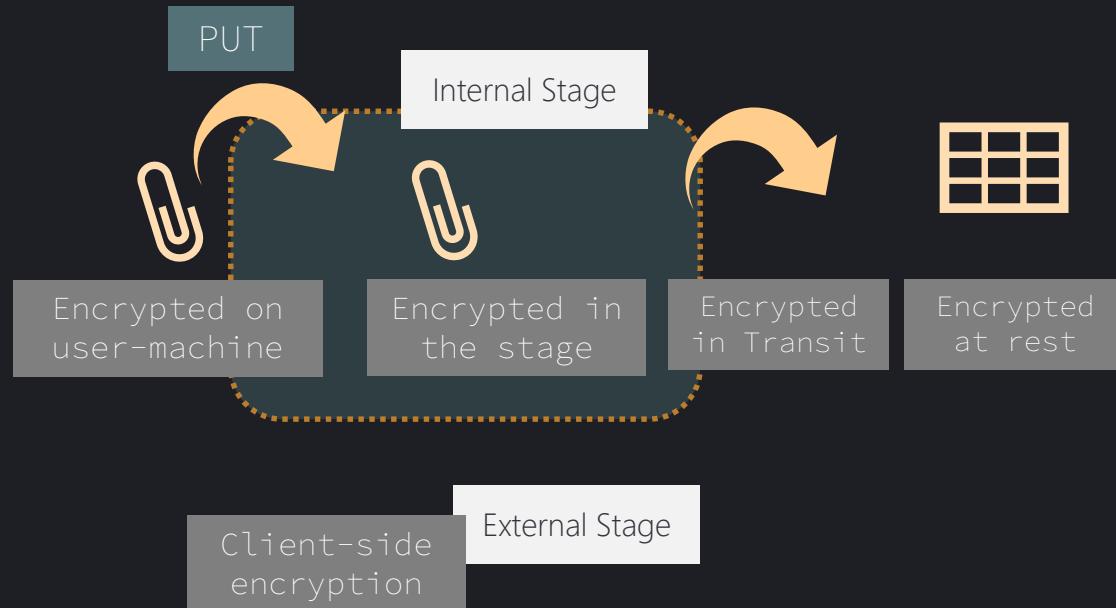
1760 0009-14563.7

73273



End-to-End Encryption

All data is encrypted at rest and in transit





Tri-Secret Secure

Enables customer to use own keys

Business Critical Edition

Snowflake Support



Customer-managed

E.g. Azure Key Vault

Snowflake-managed

Master Key

Composite Key



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Account Usage & Information Schema



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Account Usage and Information Schema

Query object metadata and historical usage data



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Account Usage

- ✓ ⏺ SNOWFLAKE
 - > ⏺ ACCOUNT_USAGE
 - > ⏺ ALERT
 - > ⏺ BCR_ROLLOUT
 - > ⏺ CORE
 - > ⏺ DATA_SHARING_USAGE
 - > ⏺ INFORMATION_SCHEMA
 - > ⏺ ORGANIZATION_USAGE
 - > ⏺ READER_ACCOUNT_USAGE



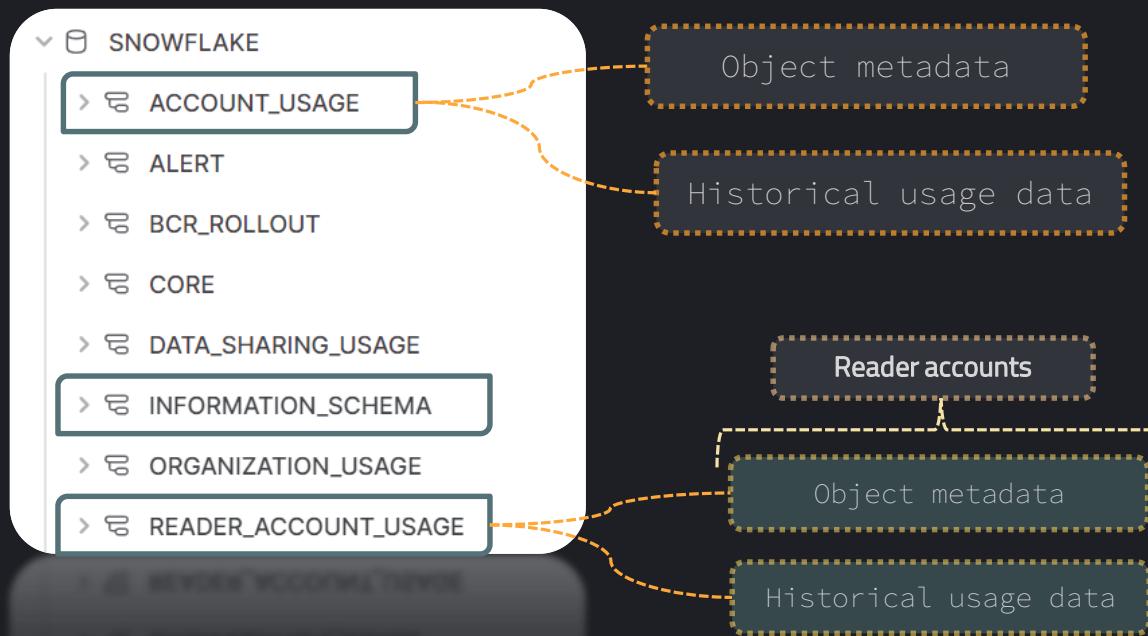
Shared database

ACCOUNTADMIN
can view everything





Account Usage



003-1040559

1250 003-77156.8

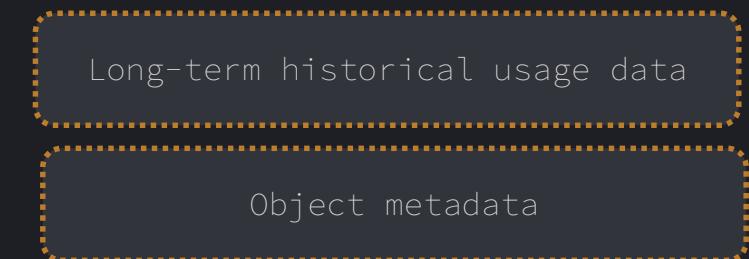
1760 0009-14563.7

73273



Account Usage

- SNOWFLAKE
 - ACCOUNT_USAGE
 - Views
 - ACCESS_HISTORY
 - ALERT_HISTORY
 - AUTOMATIC_CLUSTERING...
 - COLUMNS
 - COMPLETE_TASK_GRAPHS
 - COPY_HISTORY
 - DATABASES
 - DATABASE_REPLICATION...



COLUMN_ID	COLUMN_NAME	TABLE_ID	TABLE_NAME
1	JOB	16386	TEST
2	FIRST_NAME	18444	SEQUENCE_TEST
3	ORDER_ID	2054	ORDERS

FILE_NAME	STAGE_LOCATION
/OrderDetails_error.csv	s3://bucketsnowflakes4
Orders2.csv	s3://snowflakebucket-copyoption/returnfailed/
OrderDetails_error2 - Copy.csv	s3://snowflakebucket-copyoption/returnfailed/

Reader accounts

- READER_ACCOUNT_USAGE
 - Views
 - LOGIN_HISTORY
 - QUERY_HISTORY
 - RESOURCE_MONITORS
 - STORAGE_USAGE
 - WAREHOUSE_METERING...



Account Usage

- ▼ SNOWFLAKE
- ▼ ACCOUNT_USAGE
 - ▼ Views
 - ACCESS_HISTORY
 - ALERT_HISTORY
 - AUTOMATIC_CLUSTERING...
 - COLUMNS
 - COMPLETE_TASK_GRAPHS
 - COPY_HISTORY
 - DATABASES
 - DATABASE_REPLICATION...

Long-term historical usage data

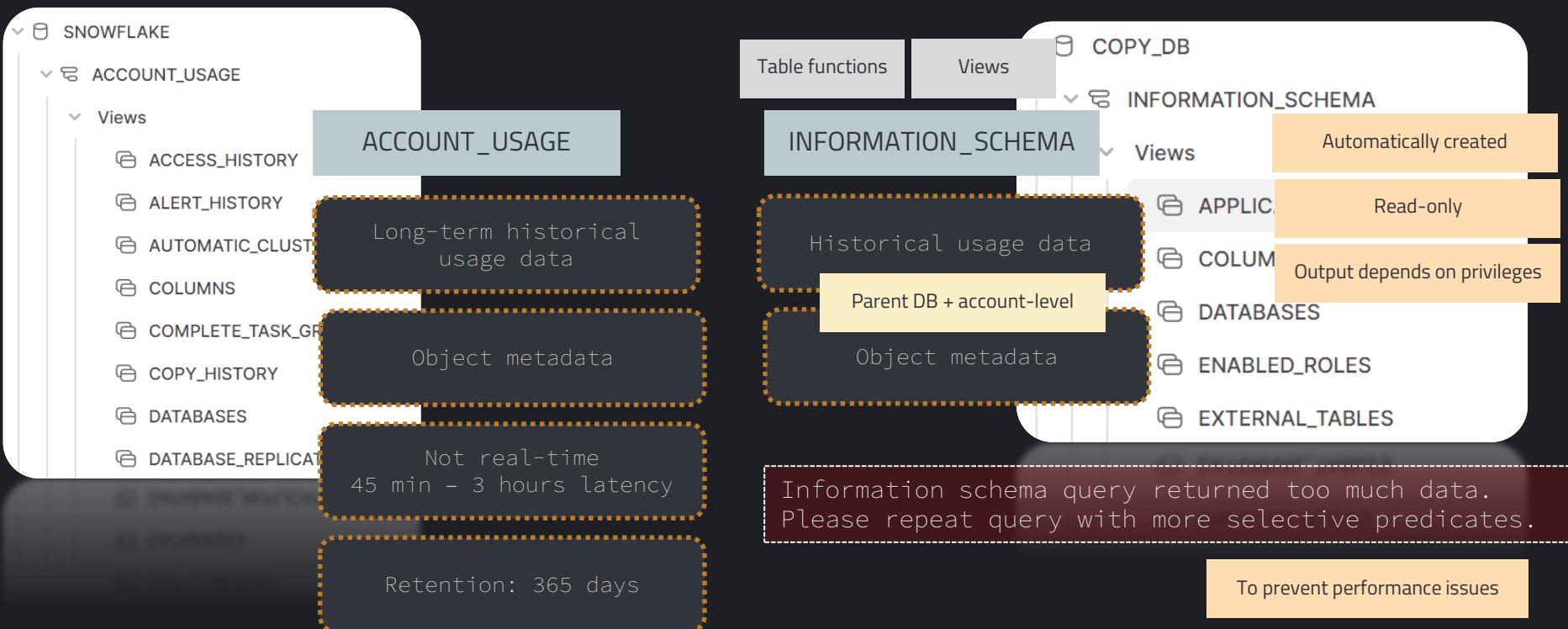
Object metadata

Data provided is not real-time
45 min - 3 hours latency

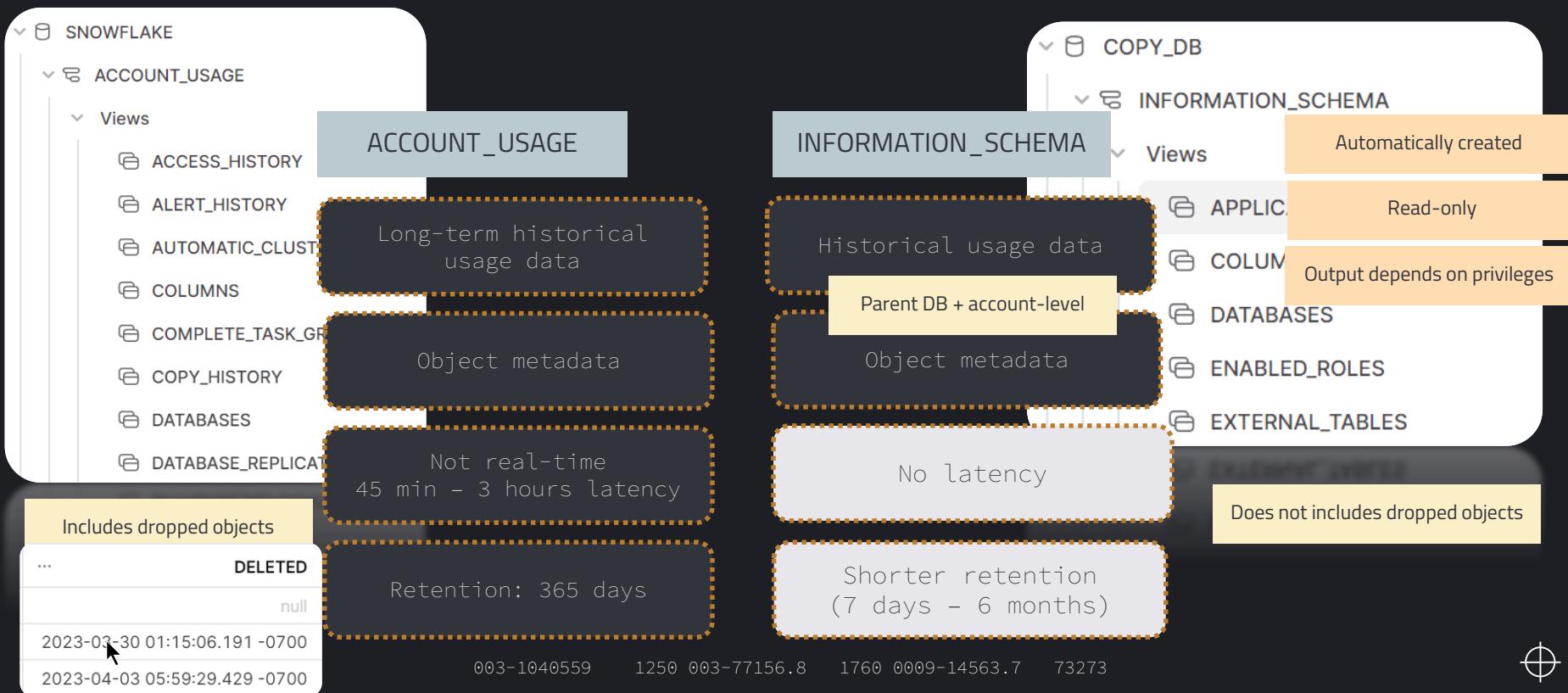
Retention: 365 days



Account Usage vs. Information Schema



Account Usage vs. Information Schema





Release Process



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





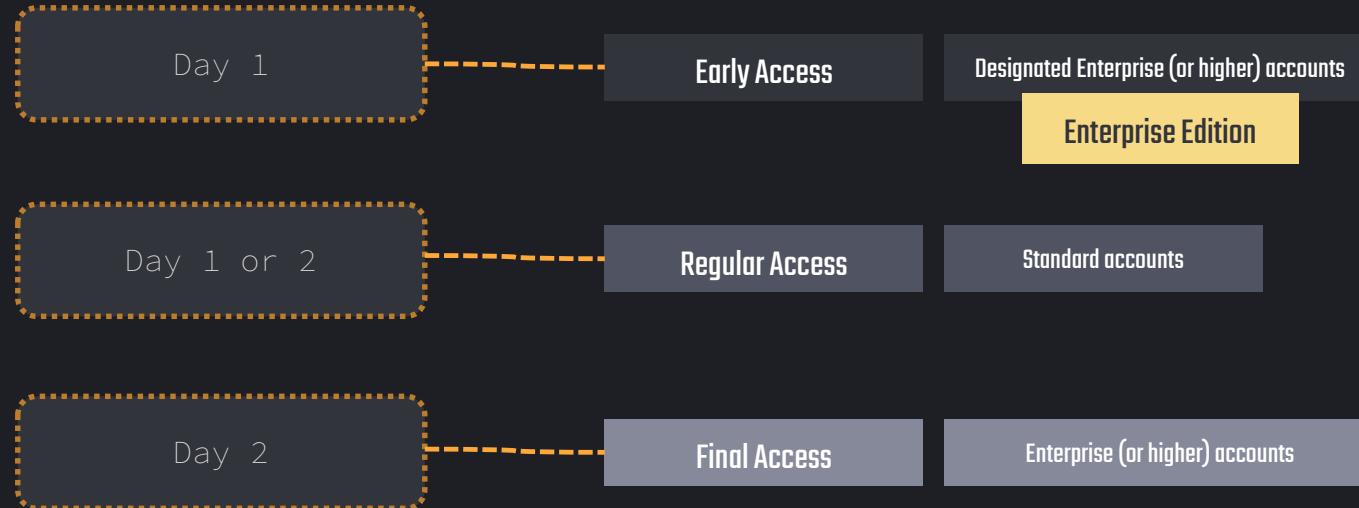
Releases

How are releases deployed?



Releases: Three stage approach

Helps to monitor and react to issues





Performance Concepts

Domain 3.0:
Performance Concepts





Query Profile & History



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Query Profile

Provide execution details for a query



003-1040559

1250 003-77156.8

1760 0009-14563.7

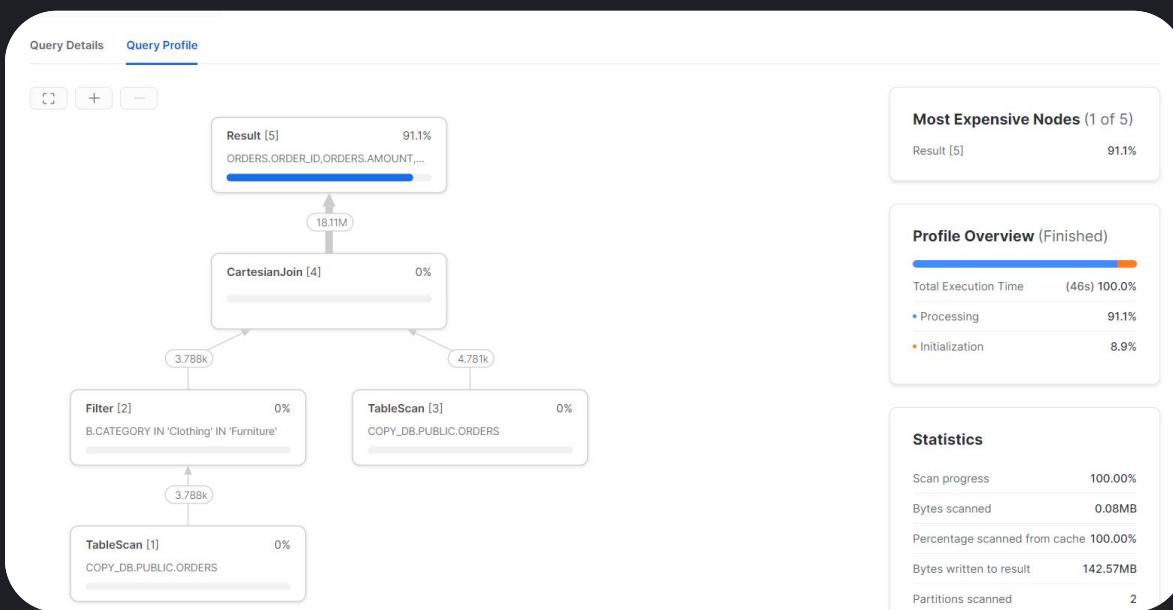
73273





Query Profile

Provide execution details for a query



003-1040559

1250 003-77156.8

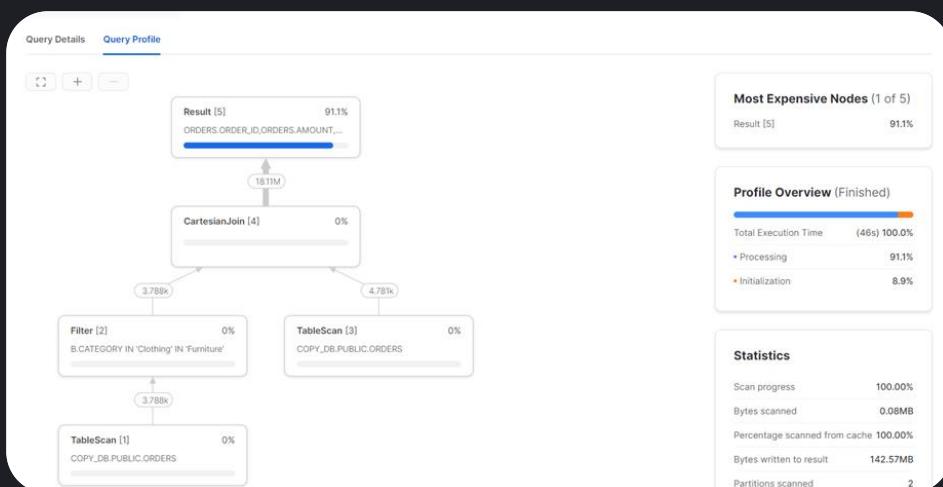
1760 0009-14563.7

73273



Query Profile

Provide execution details for a query



When to use?

Understand mechanics of a query

Performance and behavior of a query

Identify performance bottlenecks



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Query Profile

Provide execution details for a query

Available for all queries (Completed, Failed, Running)

SQL TEXT	QUERY ID	STATUS
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPC...	01ab7153-0001-0b68-0003-a4c6000412...	Running
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPC...	01ab7150-0001-0b68-0003-a4c6000412...	Success
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPC...	01ab714a-0001-0b92-0003-a4c60003f1e6	Success
SELECT * FROM SNOWFLAKE_SAMPLE_DATA.TPC...	01ab713b-0001-0b92-0003-a4c60003f186	Failed

Query Details **Query Profile**

```
graph TD; TS1[TableScan [1]] --> F1[Filter [2]]; TS1 -- "3.788k" --> F1; F1 --> TS2[TableScan [3]]; F1 -- "3.788k" --> TS2; TS2 -- "4.781k" --> CJ[CartesianJoin [4]]; CJ -- "3.788k" --> R1[Result [5]]; CJ -- "18.11M" --> R1;
```

Statistics

Scan progress	100.00%
Bytes scanned	0.08MB
Percentage scanned from cache	100.00%
Bytes written to result	142.57MB
Partitions scanned	2



003-1040559

1250 003-77156.8

1760 0009-14563.7

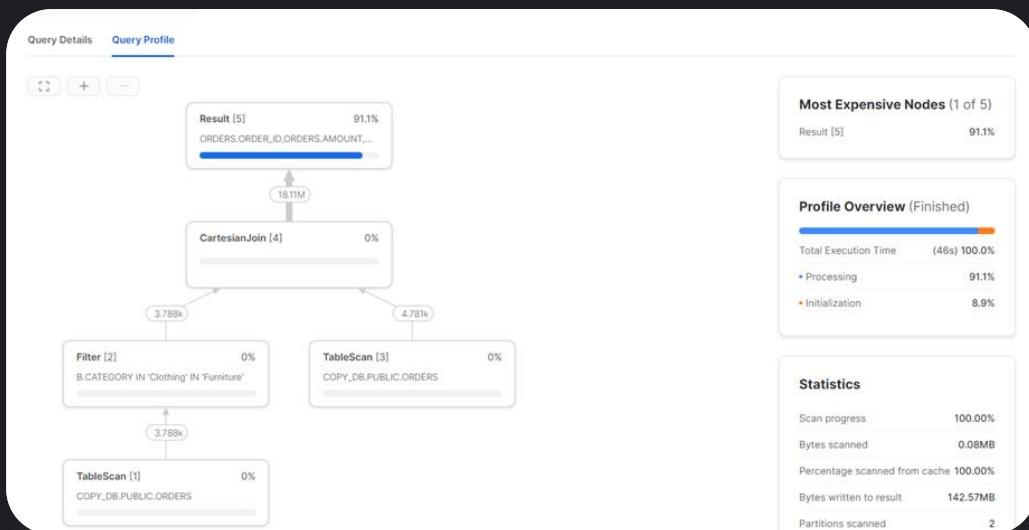
73273





Query Profile

Provide execution details for a query



Activity

Query History

Copy History
Task History

SELECT * FROM copy_db.public.orders JOIN ...
SELECT * FROM copy_db.public.orders WHERE ...

SQL Text
Query ID: 01ab710e-0001-0b68-0003-a4c6000

SUBCATEGORY

Hankerchief
Electronic Game

Result limit exceeded
Only 10,000 rows of the results are displayed. Please download the results for all of the rows.

Query Details

Query duration 142.57MB

Copy Query ID

Rows

View Query Profile



003-1040559

1250 003-77156.8

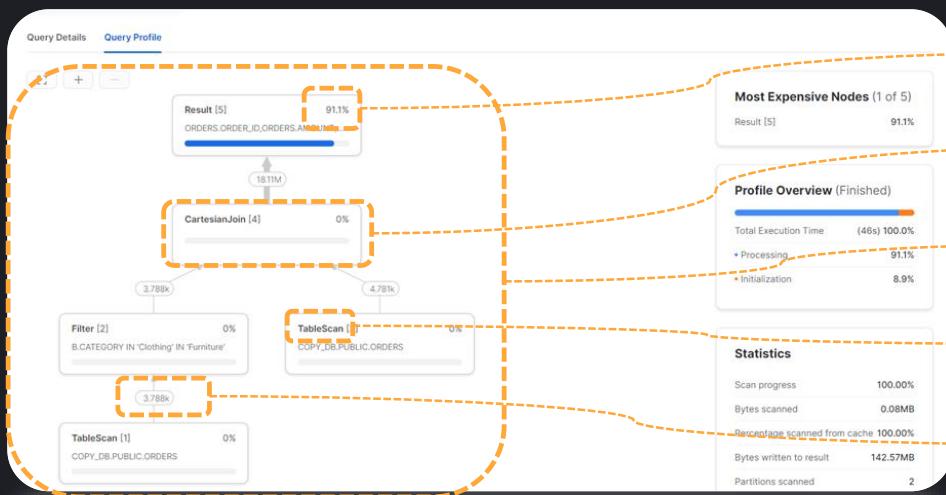
1760 0009-14563.7

73273



Query Profile

Provide execution details for a query



Percentage

Percentage of time this operator needed

Nodes

Building blocks

Operator Tree

Graphical representation

Operator Types

Aspect of query processing

Data Flow

No. of records processed



003-1040559

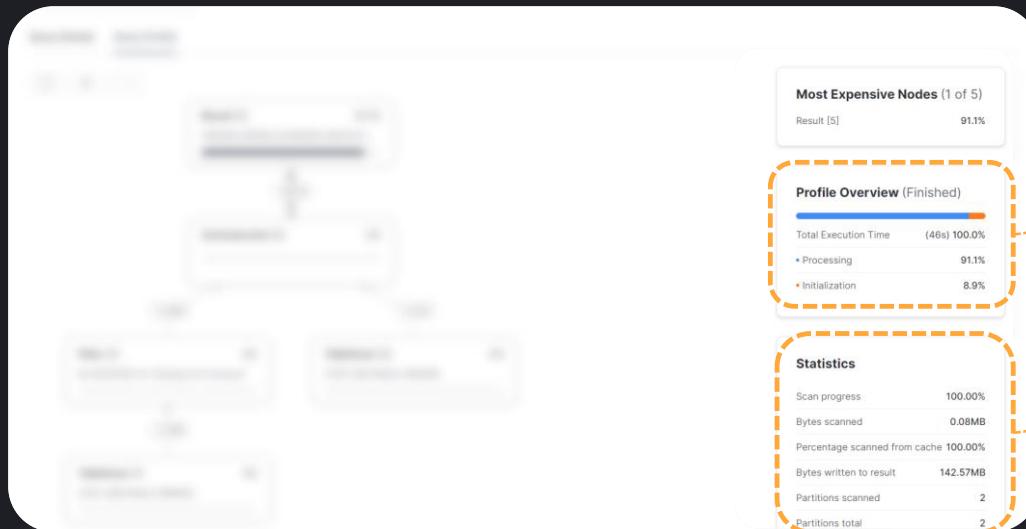
1250 003-77156.8

1760 0009-14563.7 73273



Query Profile

Provide execution details for a query



Overview

Where time was spent

Statistics

Bytes Scanned
Scanned from Cache
Data Spilling



003-1040559

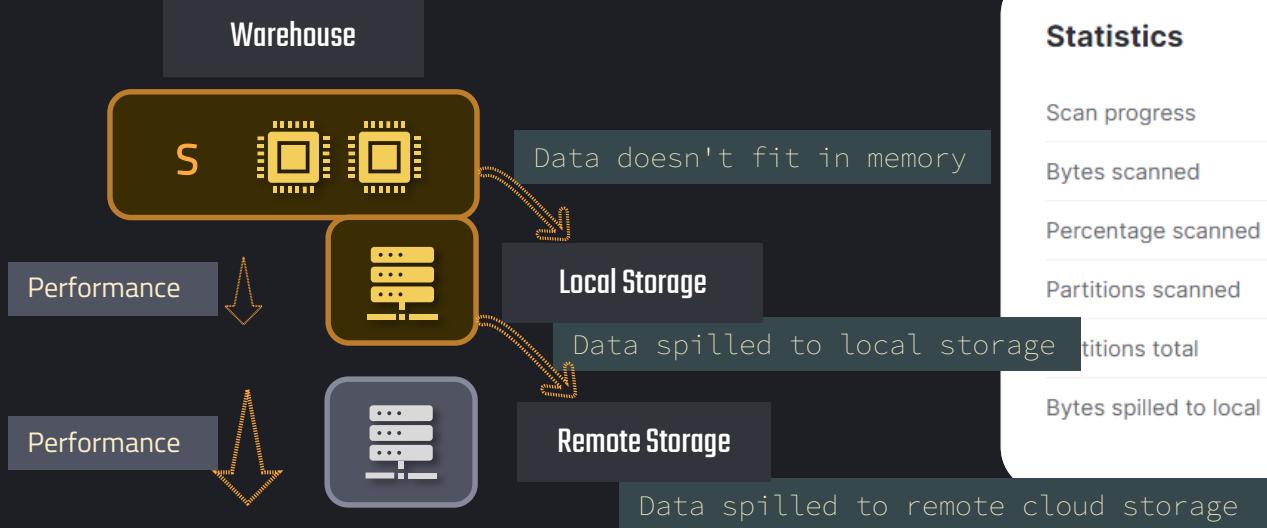
1250 003-77156.8

1760 0009-14563.7

73273



What is spilling?

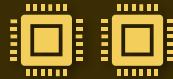




How to avoid it?

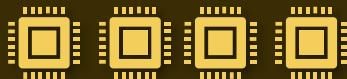
Warehouse

S



Reduce the amount of data processed

M



Increase size of warehouse





Query History

Query History in
Snowsight

The screenshot shows a dropdown menu titled 'Activity' with four options: 'Activity', 'Query History', 'Copy History', and 'Task History'. The 'Query History' option is highlighted with a blue background and a hand cursor icon.

SELECT percentile_cont(0.4) WITHIN GROUP (UNBOUNDED PRECEDING, UNBOUNDED FOLLOWING) AS median FROM table_name;

SELECT percentile_cont(0.4) WITHIN GROUP (UNBOUNDED PRECEDING, UNBOUNDED FOLLOWING) AS median FROM table_name;

SELECT percentile_cont(0.4) WITHIN GROUP (UNBOUNDED PRECEDING, UNBOUNDED FOLLOWING) AS median FROM table_name;

QUERY_HISTORY
table function in
INFORMATION_SCHEMA

```
select * from table(information_schema.query_history())  
order by start_time;
```

QUERY_HISTORY
view in
ACCOUNT_USAGE schema

```
select * from snowflake.account_usage.query_history
```

003-1040559 1250 003-77156.8

1760 0009-14563.7 73273





Caching



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Caching

CLOUD SERVICES

Statistics for tables and columns



Cached copy of the query result

QUERY PROCESSING (COMPUTE)

Virtual Warehouse



Virtual Warehouse



Virtual Warehouse

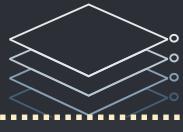
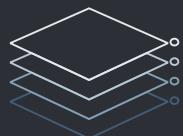


Virtual Warehouse Cache

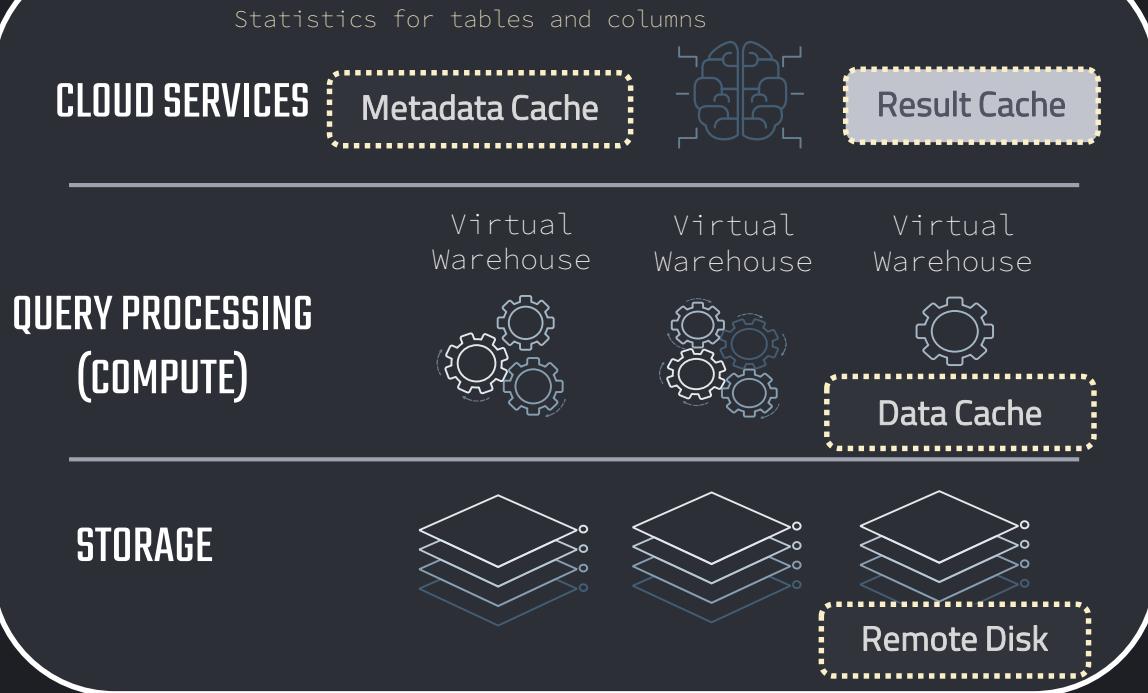
Locally caches data of the query

Local Disc I/O

STORAGE



Caching



Result Cache

Stores the results of a query (Cloud Services)

Same queries can use that cache in the future

Table data has not changed

Micro-partitions have not changed

Query doesn't include UDFs or external functions

Sufficient privileges & results are still available

Very fast result (persisted query result)

Avoids re-execution

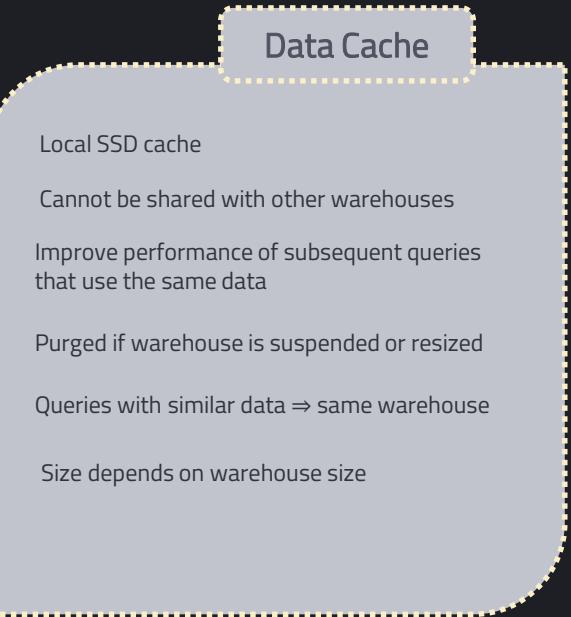
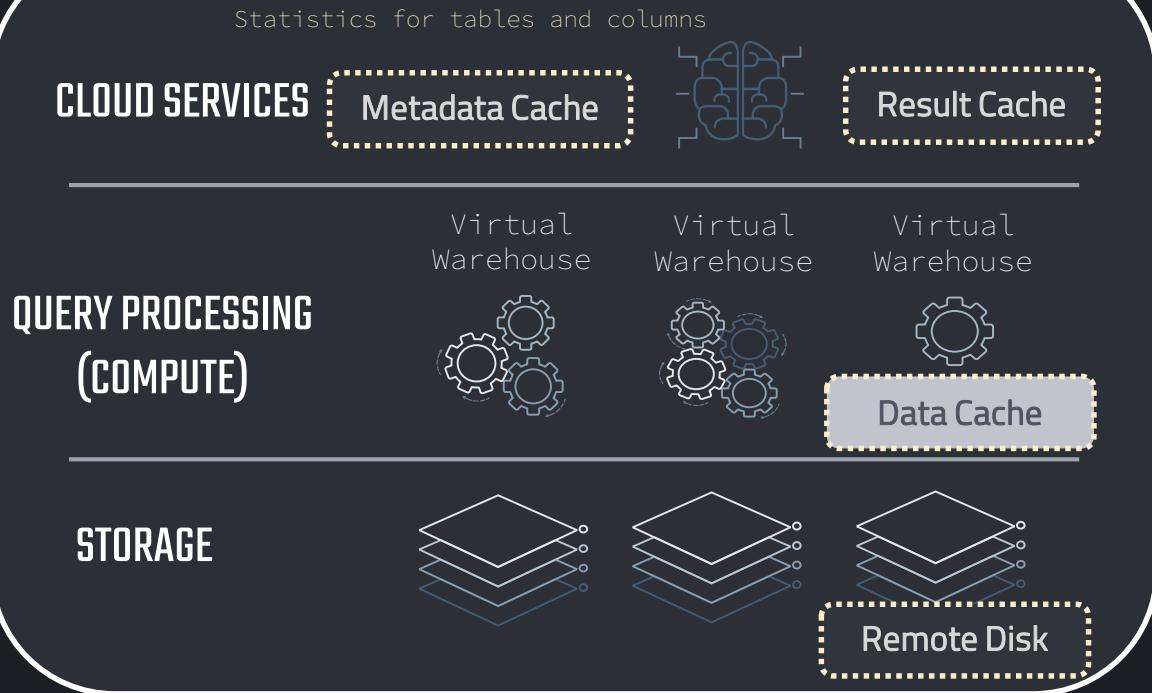
Can be disabled by using

`USE_CACHED_RESULT` parameter

If query is not re-used purged after *24 hours*

If query is re-used can be stored up to *31 days*

Caching



Caching

CLOUD SERVICES

Metadata Cache



Result Cache

QUERY PROCESSING (COMPUTE)

Virtual
Warehouse



Virtual
Warehouse

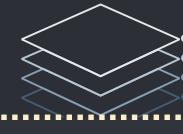


Virtual
Warehouse



Data Cache

STORAGE



Remote Disk

Metadata Cache

Stores statistics and metadata about objects

Properties for query optimization and processing

Range of values in micro-partition

Count rows, count distinct values, max/min value

Without using virtual warehouse

DESCRIBE + system-defined functions

"Metadata store"

Virtual Private Edition: Dedicated metadata store



Micro-partitions



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Micro-partitions

CLOUD SERVICES



QUERY PROCESSING
(COMPUTE)

Virtual
Warehouse



Virtual
Warehouse



Virtual
Warehouse



STORAGE



003-1040559

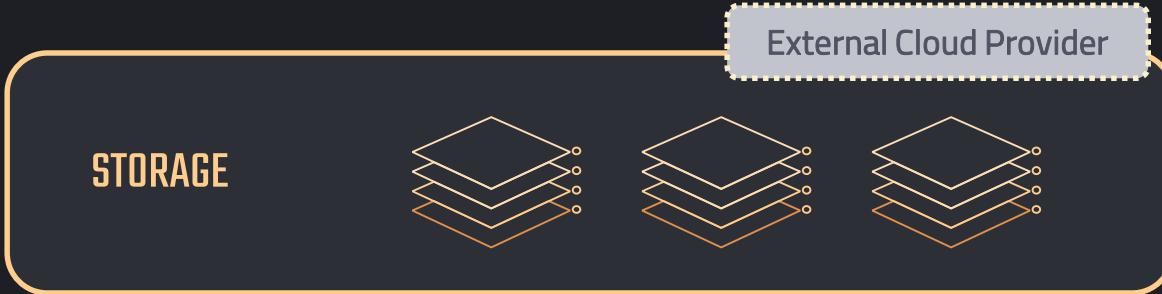
1250 003-77156.8

1760 0009-14563.7

73273



Micro-partitions



Automatically performed
Can't be disabled



Hundreds of millions

Can overlap!

Range of values

No. of distinct values

Additional properties
for query optimization



Micro-partitions

Allow very granular partition pruning

Very small

Eliminate unnecessary partitions

Contain 50-500 MB of uncompressed data

Actual size is less ⇒ data is compressed automatically

Most efficient compression algorithm found independently

Data is stored in columnar format

Unnecessary columns are eliminated when querying



003-1040559

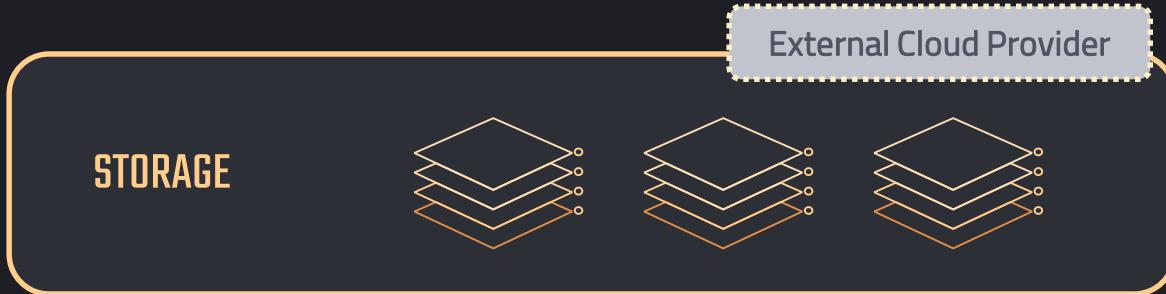
1250 003-77156.8

1760 0009-14563.7

73273



Micro-partitions



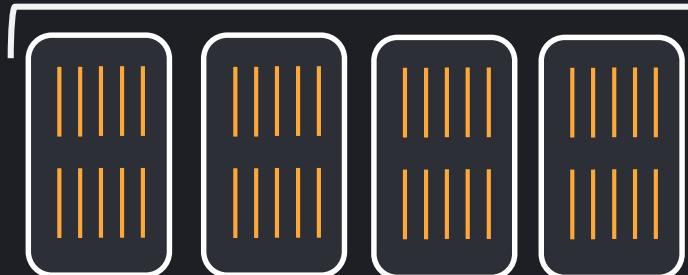
STORAGE



External Cloud Provider



Immutable
Can't be changed



Micro-partitions

new data
=
new micro-partitions
With order in which
data is created

CLUSTERING KEYS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Clustering Keys

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

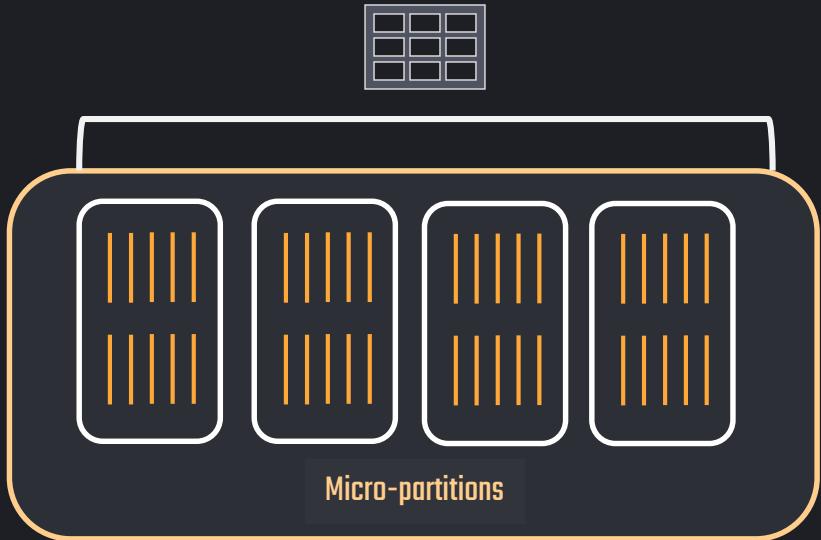


Clustering Keys





Clustering Keys



How is data stored in micro-partitions?

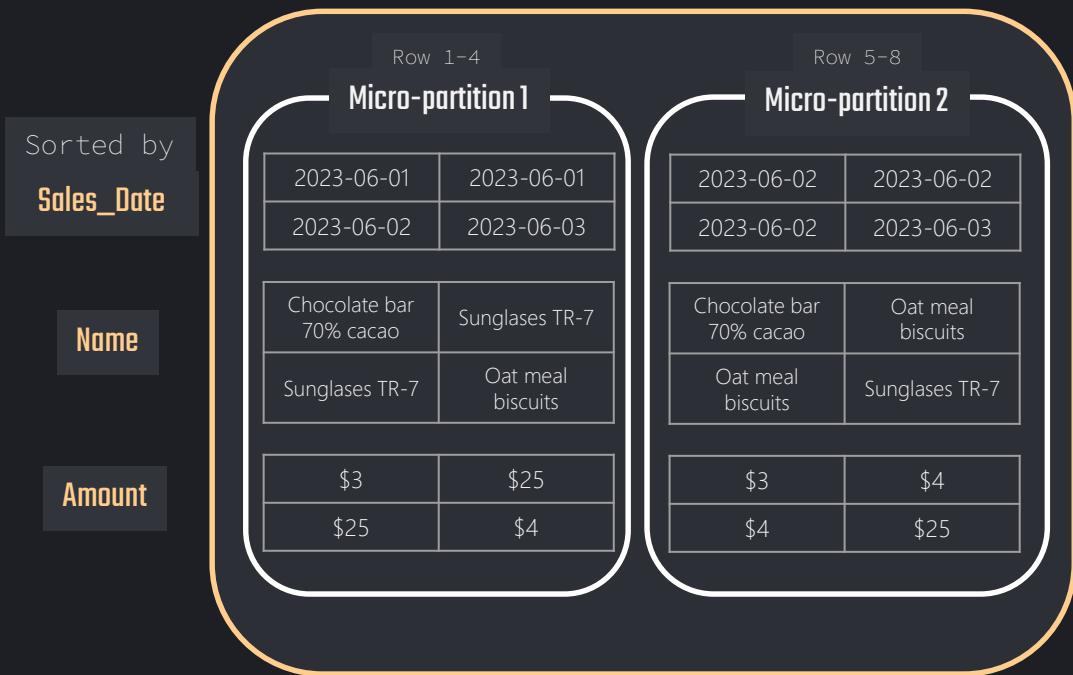
Sales_Date	Name	Amount
2023-06-02	Sunglasses TR-7	\$25
2023-06-01	Chocolate bar 70% cacao	\$3
2023-06-02	Sunglasses TR-7	\$25
2023-06-03	Oat meal biscuits	\$4
2023-06-02	Chocolate bar 70% cacao	\$3
2023-06-03	Oat meal biscuits	\$4
2023-06-02	Oat meal biscuits	\$4
2023-06-05	Sunglasses TR-7	\$25





Clustering Keys

How is data stored in micro-partitions?



Sales_Date	Name	Amount
2023-06-02	Sunglasses TR-7	\$25
2023-06-01	Chocolate bar 70% cacao	\$3
2023-06-02	Sunglasses TR-7	\$25
2023-06-03	Oat meal biscuits	\$4
2023-06-02	Chocolate bar 70% cacao	\$3
2023-06-03	Oat meal biscuits	\$4
2023-06-02	Oat meal biscuits	\$4
2023-06-05	Sunglasses TR-7	\$25



003-1040559

1250 003-77156.8

1760 0009-14563.7

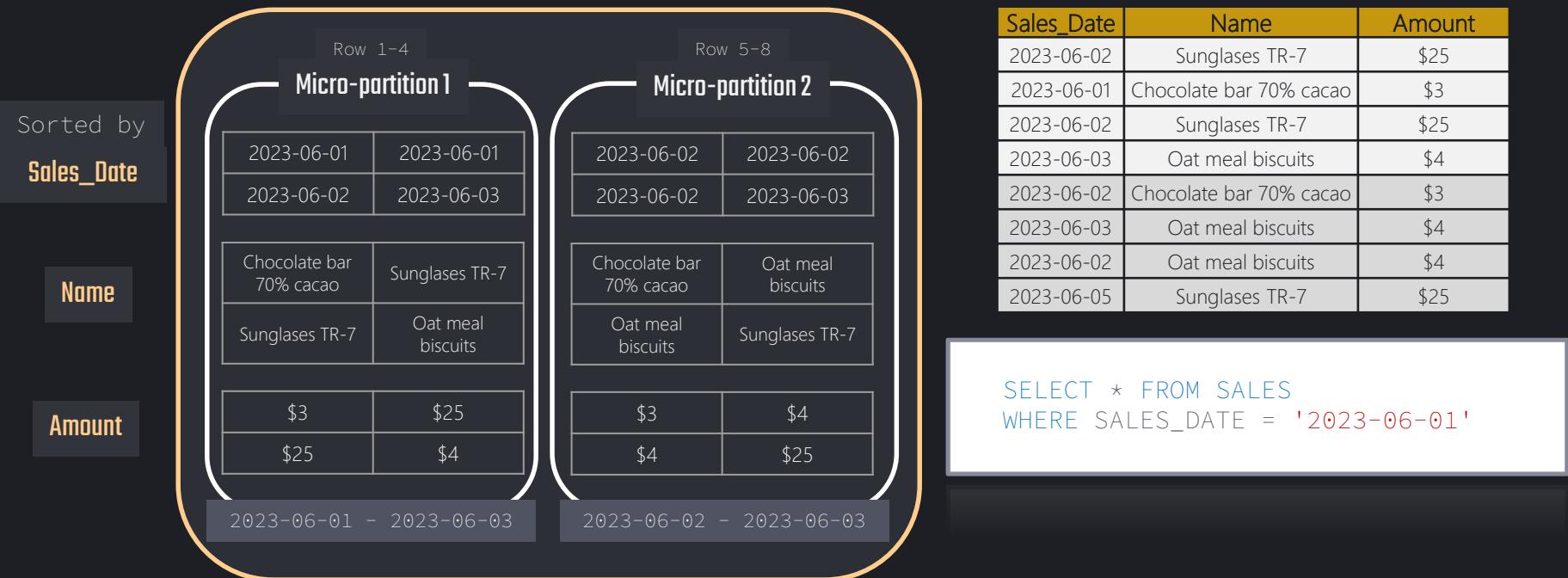
73273





Clustering Keys

How is data stored in micro-partitions?



003-1040559

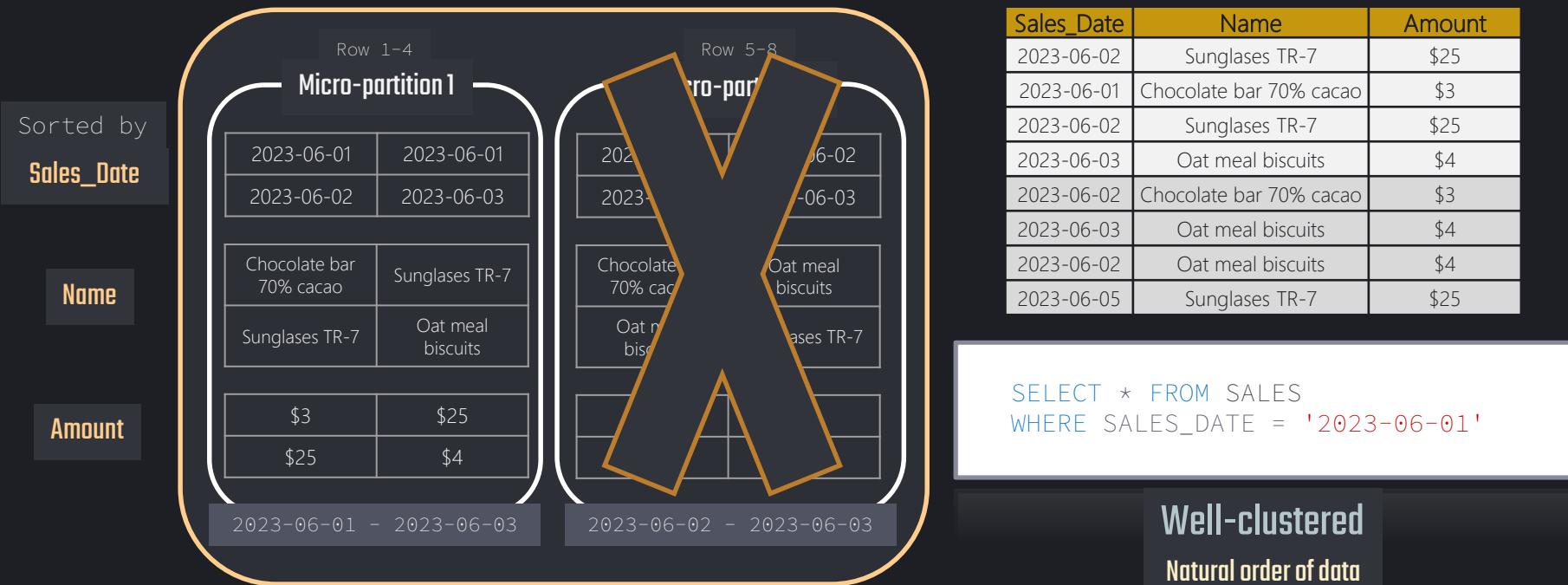
1250 003-77156.8

1760 0009-14563.7

73273

Clustering Keys

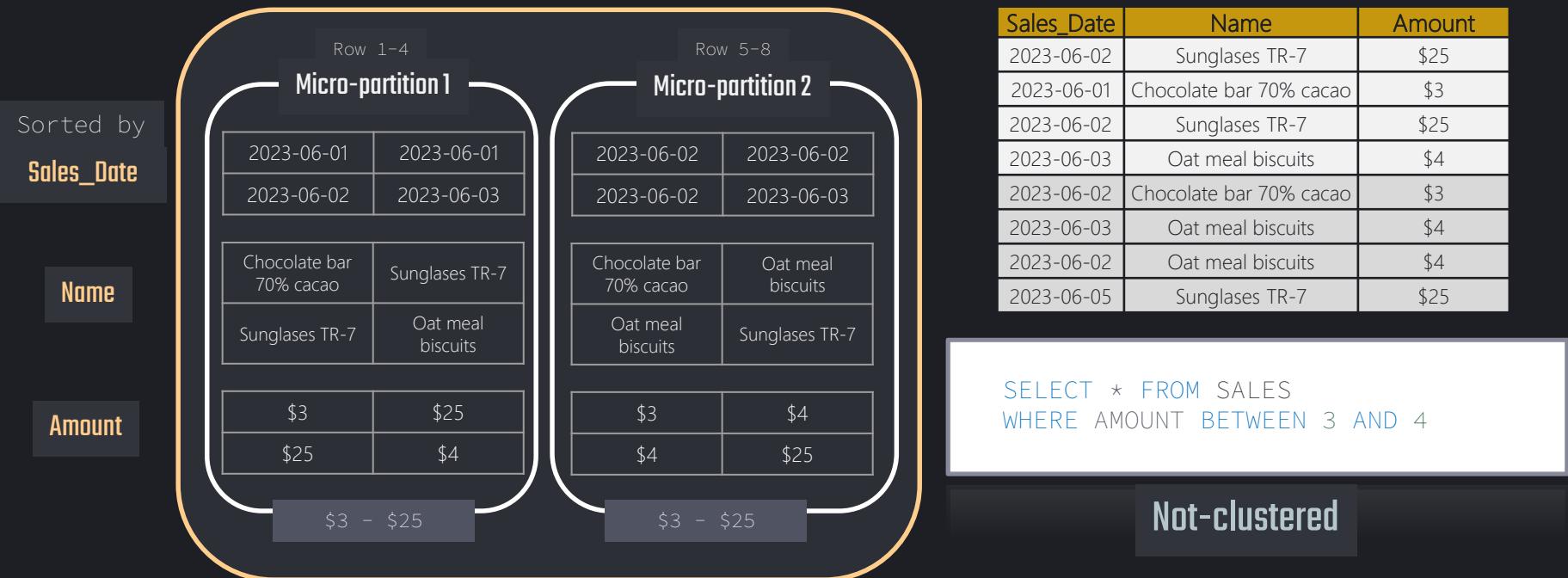
How is data stored in micro-partitions?





Clustering Keys

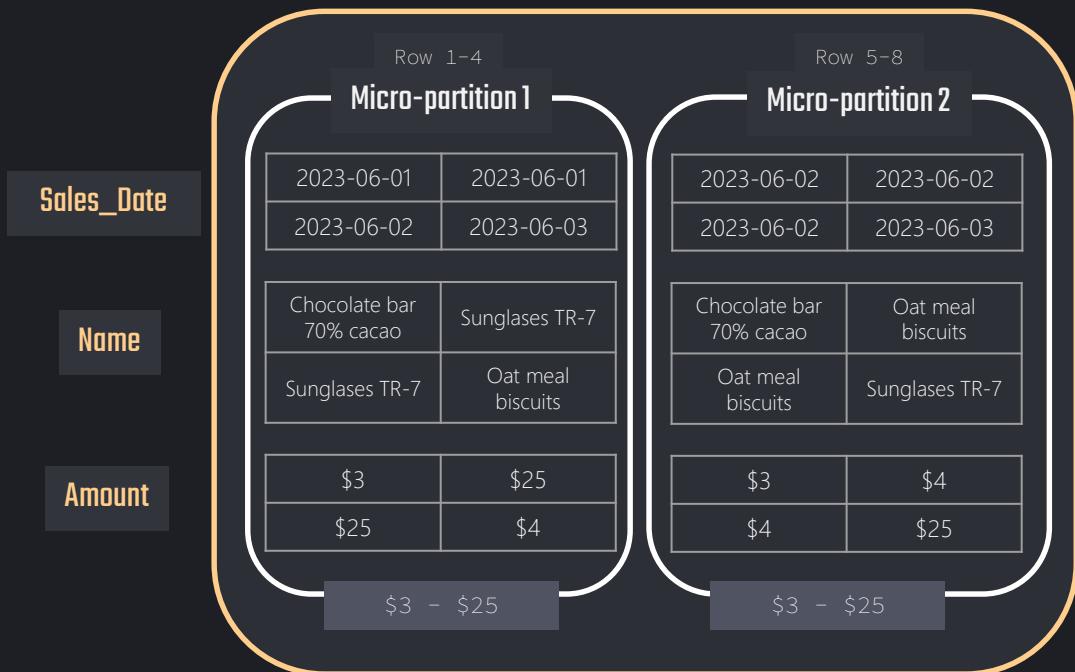
How is data stored in micro-partitions?





Clustering Keys

How is data stored in micro-partitions?



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



Clustering Keys

When is a table well-clustered?

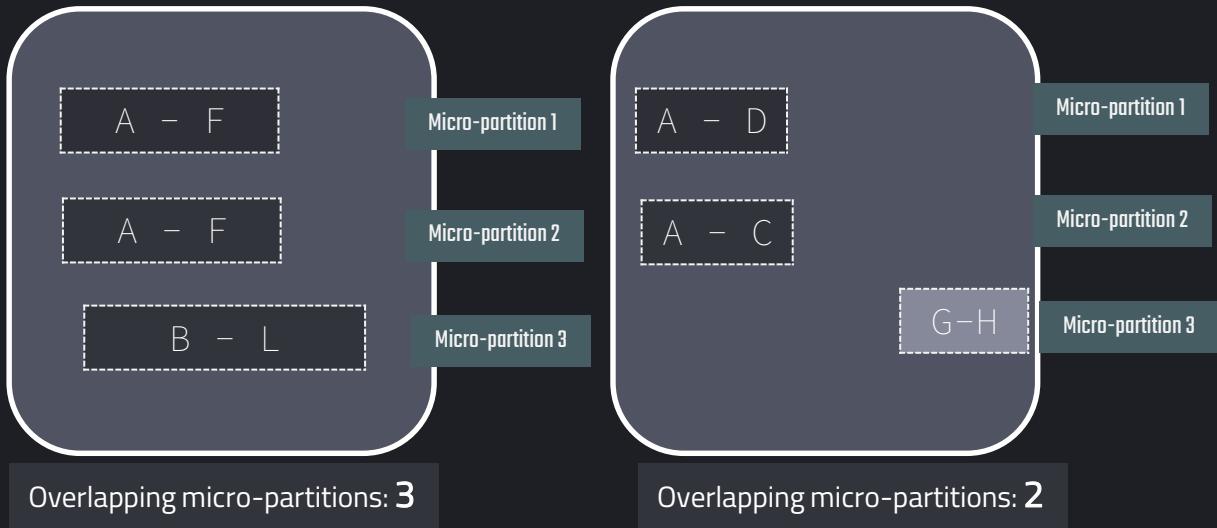
Overlapping micro-partitions

Number of partitions that overlap

Clustering depth

Average depth of the overlapping micro-partitions for specific column.

In how many micro-partitions value occurs.





Clustering Keys

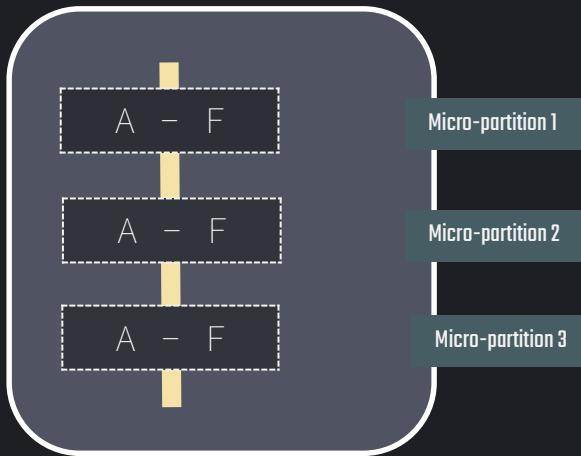
When is a table well-clustered?

Overlapping micro-partitions

Number of partitions that overlap.

Clustering depth

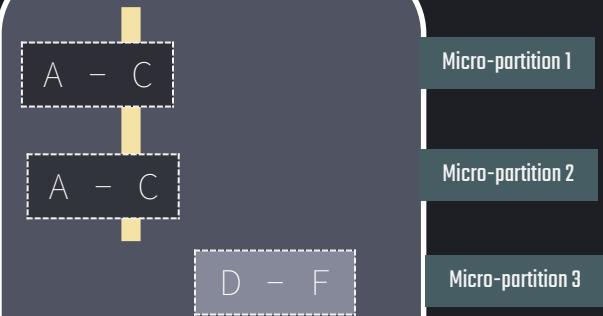
In how many micro-partitions value occurs.



Overlapping micro-partitions: 3

Average Depth: 3

Better but not ideal



Overlapping micro-partitions: 2

Average Depth: 2





Clustering Keys

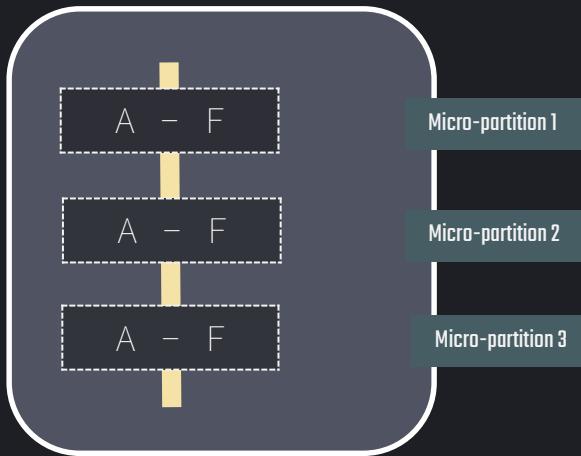
When is a table well-clustered?

Overlapping micro-partitions

Number of partitions that overlap.

Clustering depth

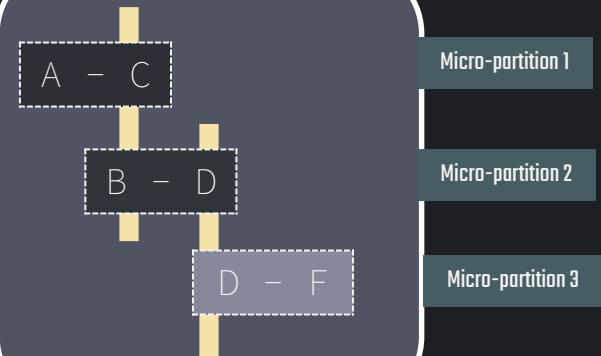
In how many micro-partitions value occurs.



Overlapping micro-partitions: 3

Average Depth: 3

Better but not ideal



Overlapping micro-partitions: 3

Average Depth: 2





Clustering Keys

When is a table well-clustered?

Overlapping micro-partitions

Number of partitions that overlap.

Clustering depth

In how many micro-partitions value occurs.

Worst



Micro-partition 1

Micro-partition 2

Micro-partition 3

Overlapping micro-partitions: 3

Average Depth: 3

Ideal

Constant state



Micro-partition 1 Micro-partition 2 Micro-partition 3

Overlapping micro-partitions: 0

Average Depth: 1



Clustering Keys

Defining Clustering Keys

More beneficial distribution of rows in micropartitions

Improved Query Performance

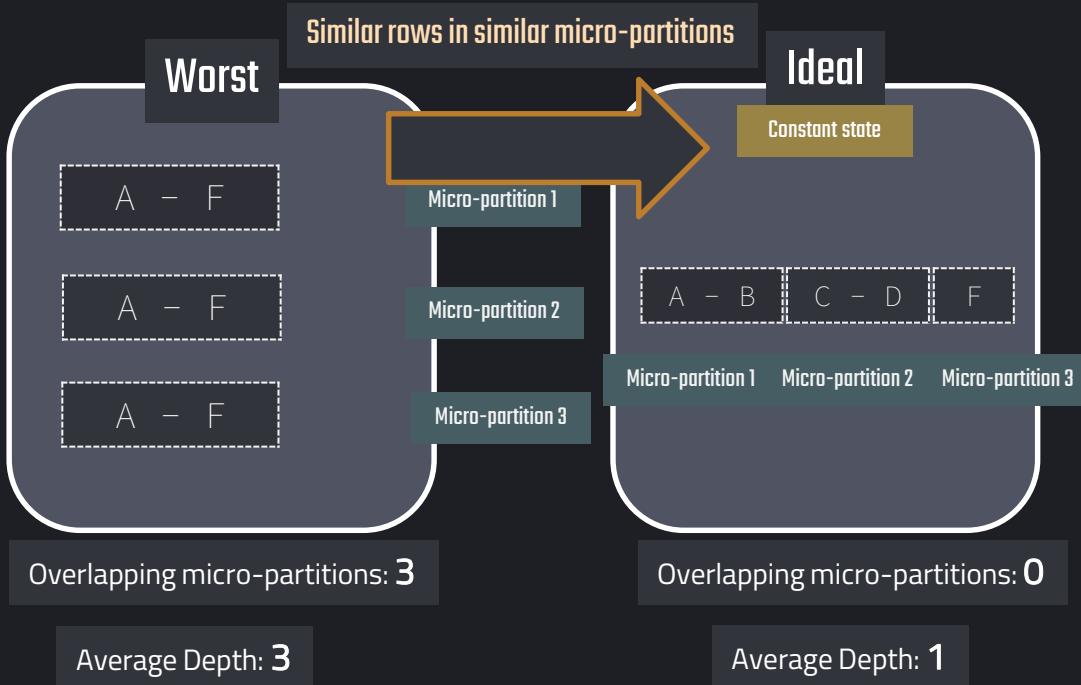
Better Scan Efficiency

Better Column Compression

Especially when columns are similar

No Future Maintenance

Fully managed by Snowflake





Defining Clustering Keys



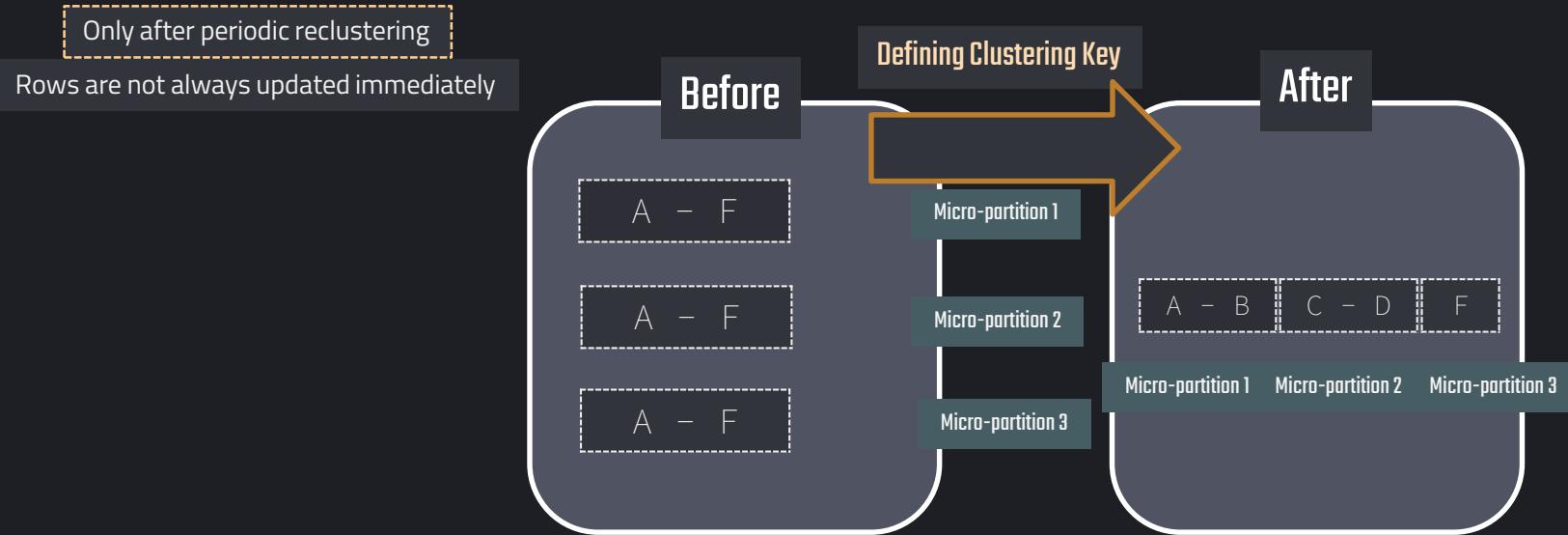
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



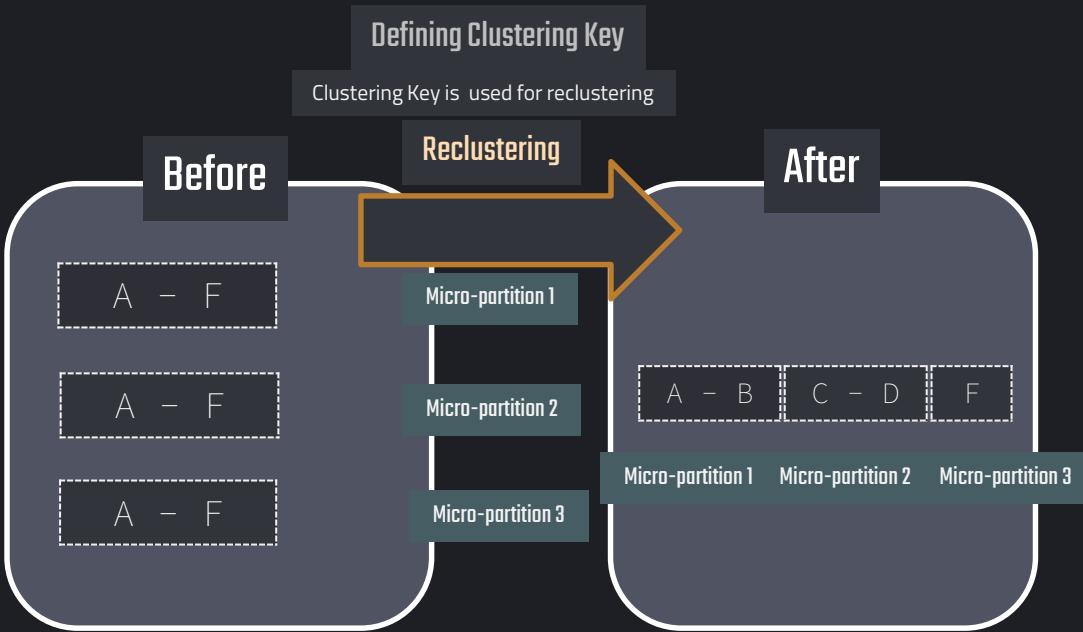
Reclustering





Reclustering

- Only after periodic reclustering
Rows are not always updated immediately
- Reclustering is automatic
Cloud Services (Serverless)
- Automatic Reclustering
Only adjusts micro-partitions that benefit



Clustering Keys

Clustering is not for all tables



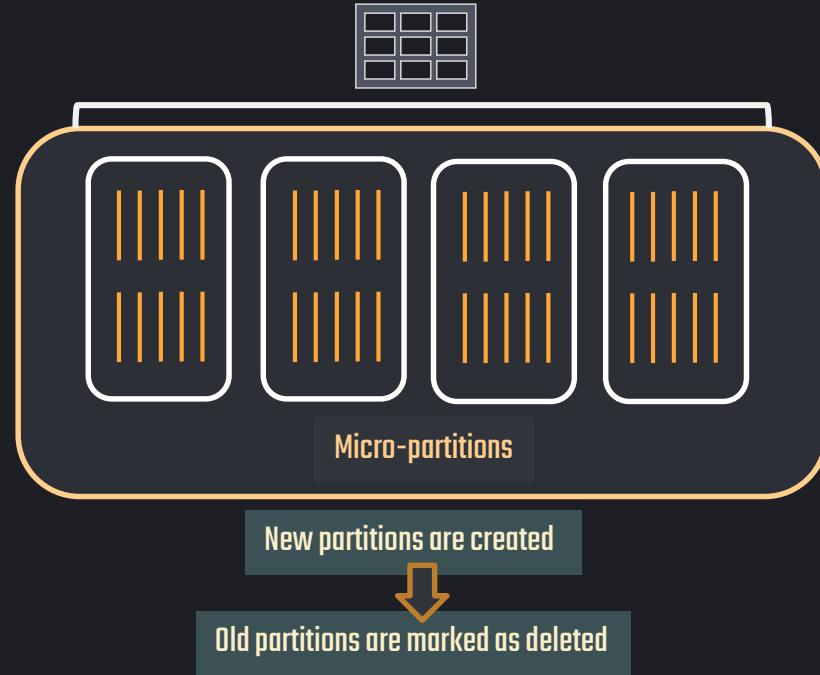
Storage Costs

Old partitions are maintained (Time Travel)



Serverless Costs

Credit consumption of reclustering



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Clustering Keys

On which columns a clustering key is most effective?

Large number of micro-partitions

Very large tables
Multiple terabytes of data.

High enough cardinality

Too low cardinality ⇒ no effective pruning

High enough cardinality

Too high cardinality ⇒ no efficient grouping

```
SELECT * FROM SALES  
WHERE AMOUNT BETWEEN 3 AND 4
```

Frequently used column in WHERE / JOIN / (ORDER BY)

Selective queries and sorting of columns
⇒ most performance improvement

Non-ideal pruning



Overhead for micro-partitioning





Clustering Keys

Clustering Key SQL commands

Adding cluster key on one or multiple columns

Cluster key can be added at any time.

```
ALTER TABLE t1 CLUSTER BY (c1, c5);
```

Low ⇒ High Cardinality

Create table with cluster key

Cluster key can be defined in table definition.

```
CREATE TABLE t1 CLUSTER BY (c1, c5);
```

Expression

Clustering key on expression.

```
ALTER TABLE t1 CLUSTER BY (DATE(timestamp));
```

Removing clustering key

Cluster key can be removed.

```
ALTER TABLE t1 DROP CLUSTER KEY;
```





System functions for Clustering Keys





System Functions on Clustering

Information on Clustering Depth

Find out more clustering information.

```
SYSTEM$CLUSTERING_INFORMATION ('table_name', ['(columns/expression)'])
```





Returning clustering information

```
SYSTEM$CLUSTERING_INFORMATION ('table_name', '(col1,col3)')
```

```
SYSTEM$CLUSTERING_INFORMATION ('table_name')
```

Not well partitioned!

```
+-----+
| SYSTEM$CLUSTERING_INFORMATION('TEST2', '(COL1, COL3)')
+-----+
{
  "cluster_by_keys" : "(COL1, COL3)",
  "total_partition_count" : 1156,
  "total_constant_partition_count" : 0,
  "average_overlaps" : 117.5484,
  "average_depth" : 64.0701,
  "partition_depth_histogram" : {
    "00000" : 0,
    "00001" : 0,
    "00002" : 3,
    "00003" : 3,
    "00004" : 4,
    "00005" : 6,
    "00006" : 3,
    "00007" : 5,
    "00008" : 10,
    "00009" : 5,
    "00010" : 7,
    "00011" : 6,
    "00012" : 8,
    "00013" : 8,
    "00014" : 9,
    "00015" : 8,
    "00016" : 6,
    "00032" : 98,
    "00064" : 269,
    "00128" : 698
  }
}
```





Returning clustering information

```
SYSTEM$CLUSTERING_INFORMATION ('CUSTOMER', '(C_NAME)')
```

	C_CUSTKEY	C_NAME
1	225,001	Customer#000225001
2	225,002	Customer#000225002
3	225,003	Customer#000225003
4	225,004	Customer#000225004
5	225,005	Customer#000225005
6	225,006	Customer#000225006
7	225,007	Customer#000225007

```
+-----+
| SYSTEM$CLUSTERING_INFORMATION('TEST2', '('CUSTOMER', '(C_NAME)')')
+-----+
{
  "cluster_by_keys" : "LINEAR(C_NAME)",
  "notes" : "Clustering key columns contain high cardinality key C_NAME which
    might result in expensive re-clustering. Consider reducing the cardinality of
    clustering keys. Please refer to https://docs.snowflake.net/manuals/user-
    guide/tables-clustering-keys.html for more information.",
  "total_partition_count" : 16,
  "total_constant_partition_count" : 16,
  "average_overlaps" : 0.0,
  "average_depth" : 1.0,
  "partition_depth_histogram" : {
    "00000" : 0,
    "00001" : 16,
    "00002" : 0,
    "00003" : 0,
    "00004" : 0,
    "00005" : 0,
    "00006" : 0,
    "00007" : 0,
    "00008" : 0,
    "00009" : 0,
    "00010" : 0,
    "00011" : 0,
    "00012" : 0,
    "00013" : 0,
    "00014" : 0,
    "00015" : 0,
    "00016" : 0
  }
}
+-----+
```





Returning clustering information

Average Depth of Table

Average Depth of Table according to specified column or clustering key.

```
SELECT SYSTEM$CLUSTERING_DEPTH ('orders', 'amount')
```

SYSTEM\$CLUSTERING_DEPTH ('ORDERS','AMOUNT')

1





Search Optimization Service



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





Search Optimization

Enterprise Edition

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path



Similar to secondary index concept

Add Search Optimization to column

Beneficial queries

Selective point look-up

Returns on one or very few rows

Equality predicates (=) or IN predicates

WHERE AMOUNT = 1

Substring and regular expression searches

e.g. LIKE or ILIKE, VARIANT column

Selective geospatial functions

with GEOGRAPHY values





Search Optimization

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path



Maintained by Search Optimization Service



Serverless Costs



Storage Costs

Credit consumption

Additional Storage needed





Search Optimization

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path

Add Search Optimization to table



```
ALTER TABLE mytable ADD SEARCH OPTIMIZATION;
```

OWNERSHIP

ADD SEARCH OPTIMIZATION privileges on schema

```
ALTER TABLE mytable DROP SEARCH OPTIMIZATION;
```





Search Optimization

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path

Add Search Optimization to table



```
ALTER TABLE mytable ADD SEARCH OPTIMIZATION ON EQUALITY (*);
```

OWNERSHIP

ADD SEARCH OPTIMIZATION privileges on schema





Search Optimization

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path

Add Search Optimization to column



```
ALTER TABLE mytable ADD SEARCH OPTIMIZATION ON GEO(mycol);
```

OWNERSHIP

ADD SEARCH OPTIMIZATION privileges on schema





Search Optimization

Can improve performance of certain types of lookup and analytical queries

Many predicates for filtering

Search Access Path

Add Search Optimization to column



```
ALTER TABLE mytable ADD SEARCH OPTIMIZATION ON GEO(mycol);
```

OWNERSHIP

ADD SEARCH OPTIMIZATION privileges on schema





Materialized Views



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





Materialized Views

Enterprise Edition

Method to handle performance issues of views

Frequently run query



View



Materialized View

SELECT ...;

Compute-intensive?



Pre-computed and physically stored



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

Performance



Updated automatically

Queries that are...

- ... frequently run
- ... sufficiently complex





Materialized Views

Method to handle performance issues of views



Serverless Costs



Storage Costs

Credit consumption

Additional Storage needed

Start slow

Resource monitors can't control Snowflake-managed warehouses

```
SELECT * FROM TABLE(INFORMATION_SCHEMA.MATERIALIZED_VIEW_REFRESH_HISTORY());
```



003-1040559 1250

003-77156.8

1760 0009-14563.7

73273



Materialized Views

Create Materialized View

Use CREATE MATERIALIZED VIEW statement

```
CREATE MATERIALIZED VIEW v_1 AS  
    SELECT * FROM table1 where c1 = 200
```

Limitations

- Query only 1 table (no joins)
- No views / materialized views
- No window functions, UDFs, HAVING
- Some aggregate functions

Can be created on external tables





Materialized Views

Create Materialized View

Use CREATE MATERIALIZED VIEW statement

```
CREATE MATERIALIZED VIEW v_1 AS  
SELECT * FROM table1 where c1 = 200
```

```
ALTER MATERIALIZED VIEW v_1 SUSPEND;  
ALTER MATERIALIZED VIEW v_1 RESUME;
```

```
DROP MATERIALIZED VIEW v_1;
```





Warehouse Considerations



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



Warehouse Considerations

Resizing

- Warehouses can be resized even when query is running or when suspended
⇒ Impact only *future* queries, not on the running one

Scale up vs. Scale out

- Scale up (resize): More complex queries
- Scale out: More users (more queries)

Dedicated warehouse

- Isolate workload of specific user
- Different type of workload ⇒ different warehouse
- Enable auto-suspend and auto-resume (available for all warehouses)

