Snowflake Creds

malston
'password'(same as friday)

https://app.snowflake.com/klnlgoz/urb84268/#/homepage

https://hub.docker.com/r/localstack/snowflake

Domain 1.0: Snowflake Cloud Data Platform Features and Architecture

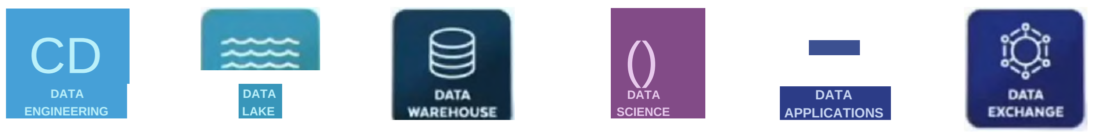
web ui

- queries can be changed to charts, which can be used in dashboards
- streamlit is tool beyond scope
-
- - copy history is data was loaded into snowflake

What is Snowflake Data Cloud?

- Charges on pay-to-go – ***consumption model***[1]
- All data reduces the risk of ***data silos***[2]
- Data science - machine learning - push logic to wear resides on platform, reduces latency
- Handles spikes in usage of application
- Sharing of data

ONE PLATFORM, ONE COPY OF DATA, MANY WORKLOADS



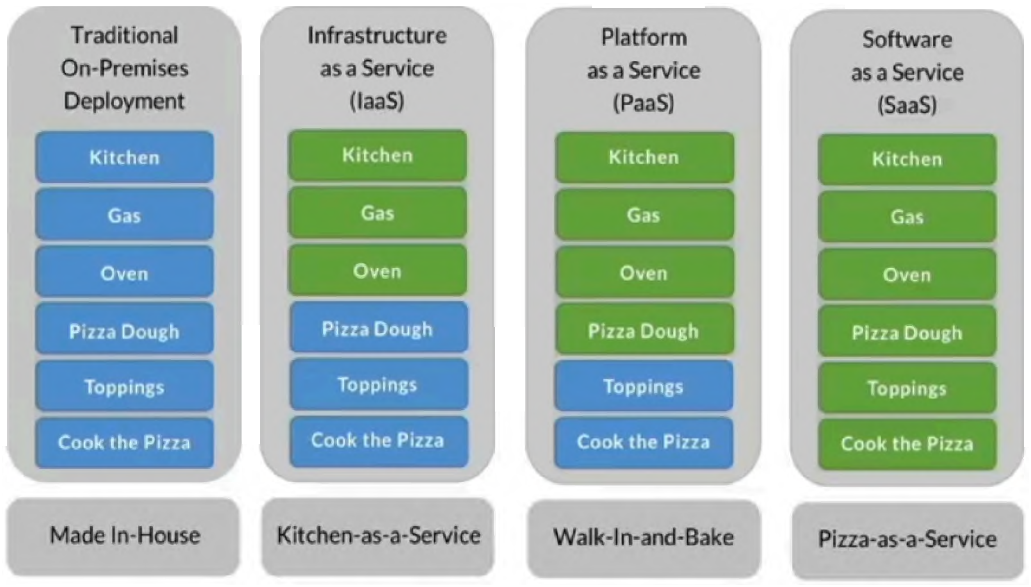
Rethink transformation with robust and integrated data pipelines	Simplify and accelerate your data take with one platform for all your data	Deliver analytics at scale with a modern data warehouse	Simplify and accelerate nachine learning and artificial intelligence	Develop apps with fast and scalable analytics that delight customers	Empower your ecosystem to securely collaborate across all data

Software as a Service

- sign up to AWS or Azure before configuring services
-

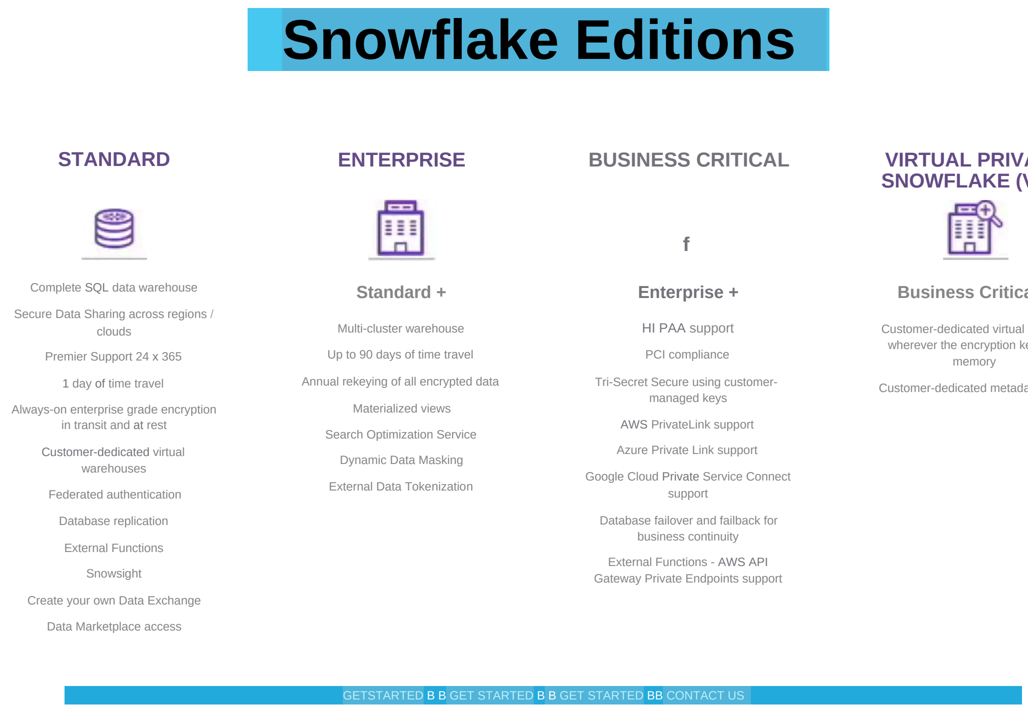
[1] consumption
[2] data silo

Software as a Service (SaaS)



- 🟦 **You Manage** 🟩 **Vendor Manages**

Snowflake Editions



Snowflake Editions

STANDARD

Complete SQL data warehouse

Secure Data Sharing across regions / clouds

Premier Support 24 x 365

1 day of time travel

Always-on enterprise grade encryption in transit and at rest

Customer-dedicated virtual warehouses

Federated authentication

Database replication

External Functions

Snowsight

Create your own Data Exchange

Data Marketplace access

ENTERPRISE

Standard +

Multi-cluster warehouse

Up to 90 days of time travel

Annual rekeying of all encrypted data

Materialized views

Search Optimization Service

Dynamic Data Masking

External Data Tokenization

BUSINESS CRITICAL

f

Enterprise +

HI PAA support

PCI compliance

Tri-Secret Secure using customer-managed keys

AWS PrivateLink support

Azure Private Link support

Google Cloud Private Service Connect support

Database failover and failback for business continuity

External Functions - AWS API Gateway Private Endpoints support

VIRTUAL PRIVATE SNOWFLAKE (VPS)

Business Critical +

Customer-dedicated virtual servers wherever the encryption key is in memory

Customer-dedicated metadata store

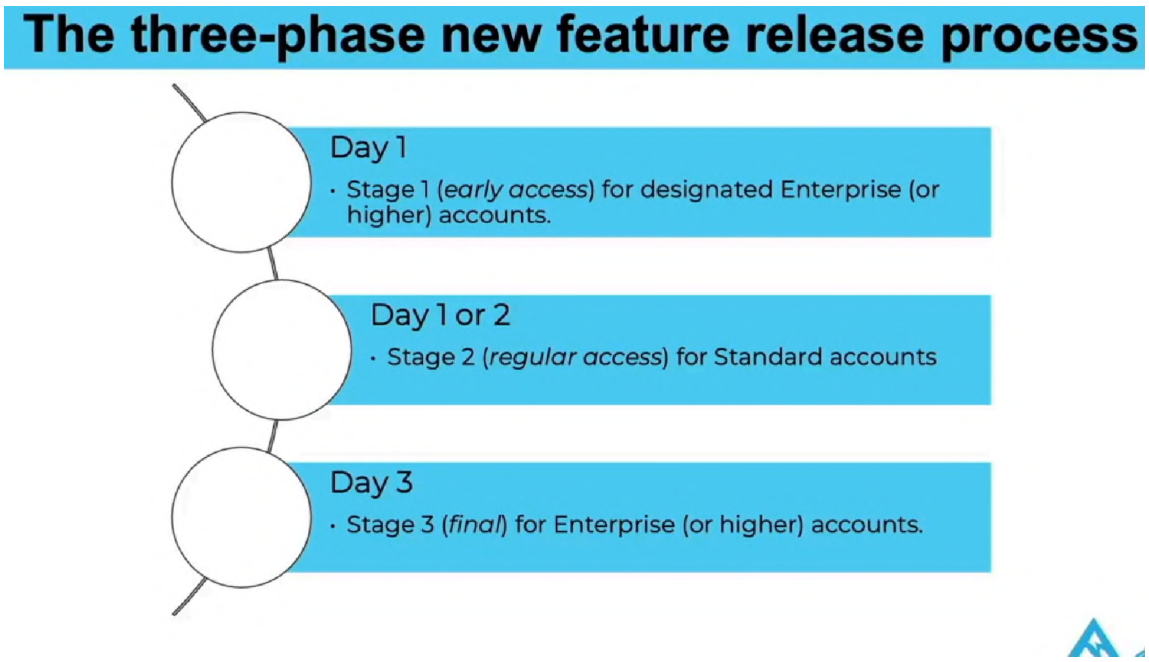
GETSTARTED B B GET STARTED B B GET STARTED BB CONTACT US

Commercial options

- Factors that influence cost
  - On-demand capacity account
  - Capacity
  - ***Typically a decision between Enterprise and Business Critical – usually opted for capacity account model***

The three-phase new feature release process

- Aims to release features weekly



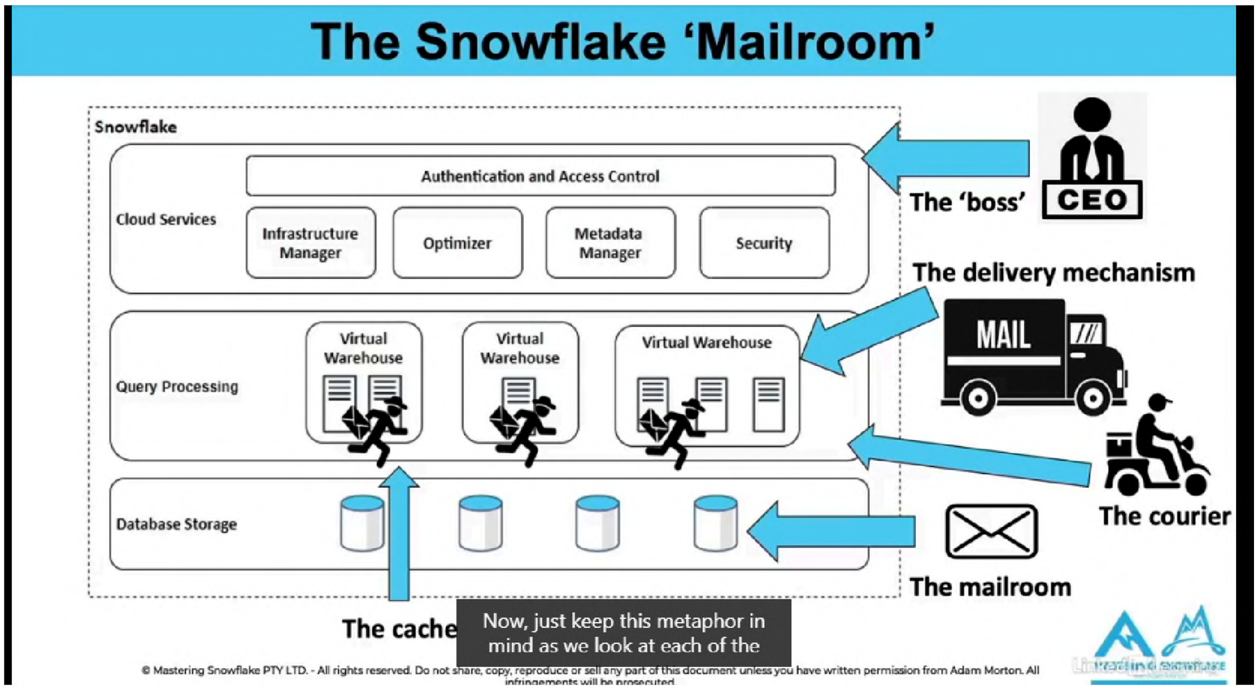
The three-phase new feature release process

Day 1
- Stage 1 (*early access*) for designated Enterprise (or higher) accounts.

Day 1 or 2
- Stage 2 (*regular access*) for Standard accounts

Day 3
- Stage 3 (*final*) for Enterprise (or higher) accounts.

The Snowflake 'Mailroom' (3 Tier architecture)



Now, just keep this metaphor in mind as we look at each of the

- Cache and storage are independent; can be

Storage Layer

Database Storage Layer

Behind the scenes, Snowflake stores data in a compressed, native format known as ***micro-partitions.***

These are small blocks of storage that are 50-500 MB uncompressed—note that once in Snowflake, they're even smaller, as all data in Snowflake is compressed.

If you have a very large table, you'll have millions of these micro-partitions scattered across the storage layer. How is data organized across them?

Well, Snowflake automatically selects a ***clustering key*** based on when the data is loaded. Metadata is collected and stored, which allows the ***query optimizer*** to understand what data is contained within each micro-partition.

AAA

Query processing layer

Query Processing Layer

The ***Query Processing*** foyer provides the resources to execute the query by using a ***virtual warehouse.***

A virtual warehouse is essentially the basket of compute (CPU) and memory (RAM) required to run pretty much any SQL data manipulation language ***(DML)*** operations against the data in Snowflake.

This also includes loading data into Snowflake tables. As we'll see in the next domain, virtual warehouses come in a range of t-shirt sizes from extra small to large, with variations in between.

Virtual warehouses can access any of the underlying data in Snowflake. You can also start, stop, drop, and create them as you need them, with no impact on any other data or operations within Snowflake.

AZ

- Cloud Services Layer

Cloud Services Layer

This layer is the head office of our postal service, the brains of the operation. This collection of services-which we break down below—ties together everything we have discussed so far.

- **Authentication—**This aspect manages the service that allows users and applications to log on to the Snowflake platform.
- **Infrastructure Management—**This aspect looks after the management of the underlying cloud infrastructure.
- **Metadata Management—**This service collects metadata when various operations are carried out on the Snowflake platform.
- **Query Parsing and Execution—** This service takes care of the ***query planning*** to work out the most efficient way to process queries. It compiles the queries to ensure no syntactical errors and manages the query execution. Snowflake uses a cost-based optimizer, which determines the fastest path to access the data by generating a range of query plans—based on the metadata available—before assessing which one is optimal to use. This is referred to as ***query optimization.***

The challenges of sharing data

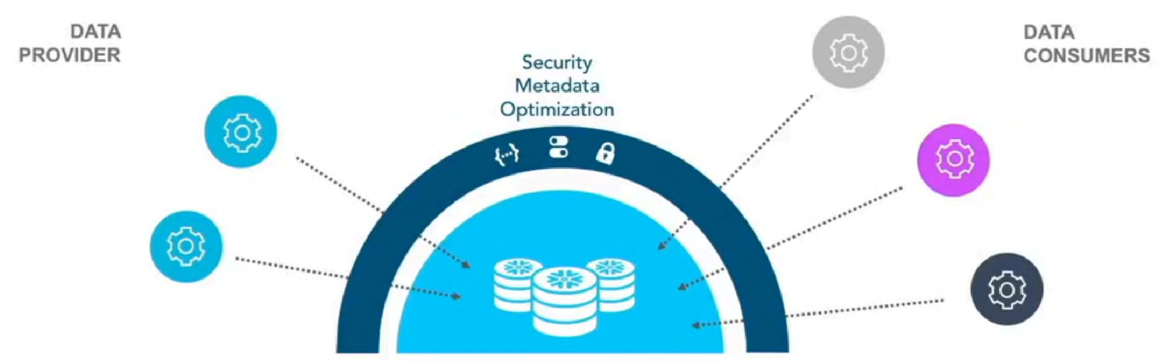
- 2

The challenges of sharing data

- APIs and SFTPs are traditional ways of sending files between organizations - and is still commonplace.
- This process has a lot of points of failure and risk
- Users still need to learn the structure of an API and how to query it before physically copying data over the network so that it can be analyzed or provided to downstream data consumers or applications.
- This means zero latency, no need for duplicate datasets or additional storage, and less risk of errors than any process that involves the physical movement of data.

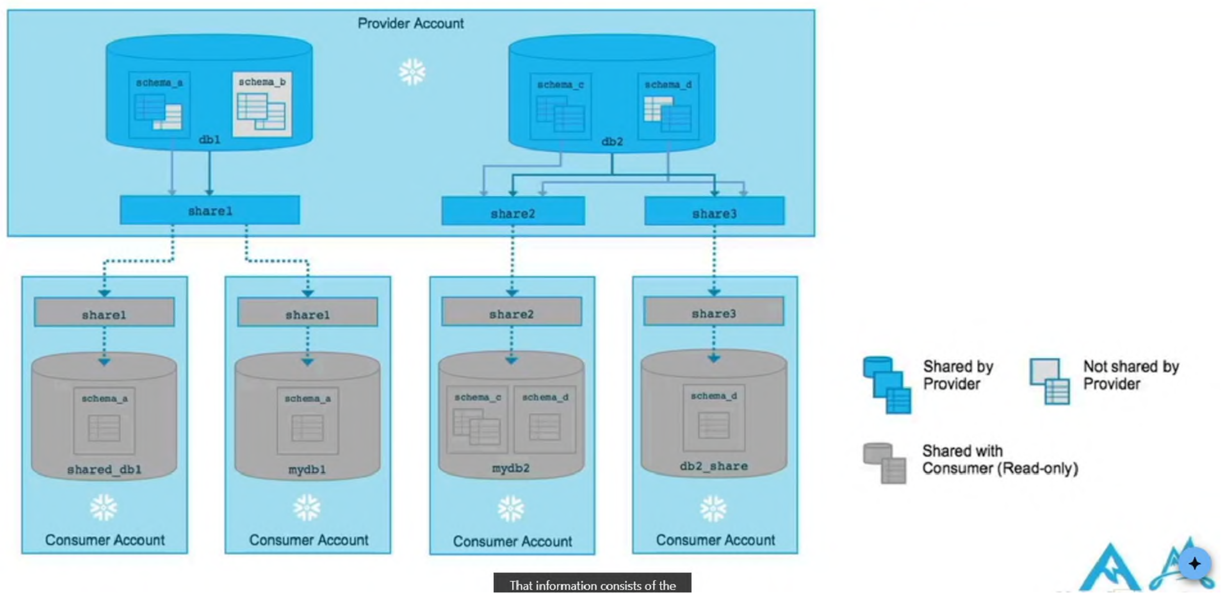
- Data Providers and Data Consumers
- 2 main types of accounts

Data Sharing Account Types

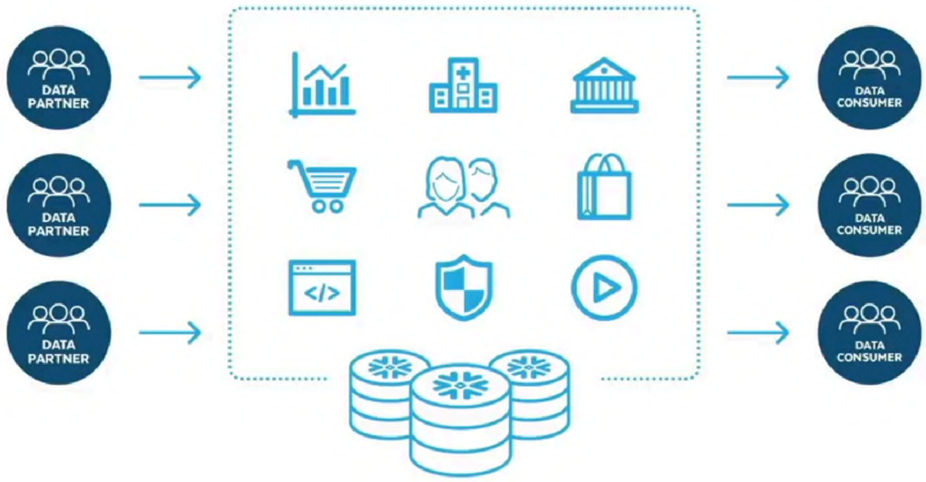


data provider - makes data available to other SF accounts to consume; granular access control functionality to manage shared resources
data consumer - creates a database from share made available by data provider

Introduction to Data Sharing



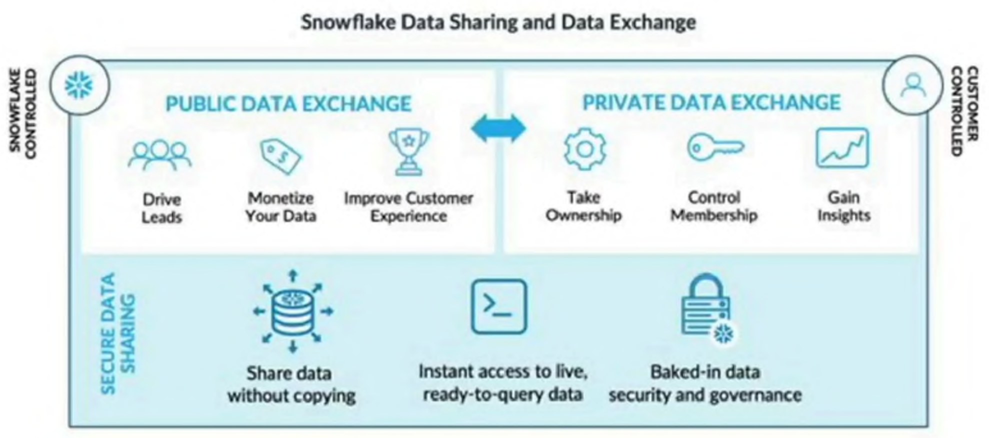
Data Sharing – Data Marketplace



You can also use the Snowflake Marketplace to discover and access

Market place to discover and access 3rd party datasets and applications – allowing to share curated data offerings to monetize; your own products/applications across the snowflake data cloud – rather than maintaining shares with each individual
Forecasting and machine learning, streaming (weather, traffic, identity data for subscribers, insights, etc.)
Private data exchange

Data Sharing – Data Exchanges


Snowflake Data Sharing and Data Exchange
SNOWFLAKE CONTROLLED
PUBLIC DATA EXCHANGE
Drive Leads
Monetize Your Data
Improve Customer Experience
PRIVATE DATA EXCHANGE
Take Ownership
Control Membership
Gain Insights
CUSTOMER CONTROLLED
SECURE DATA SHARING
Share data without copying
Instant access to live, ready-to-query data
Baked-in data security and governance

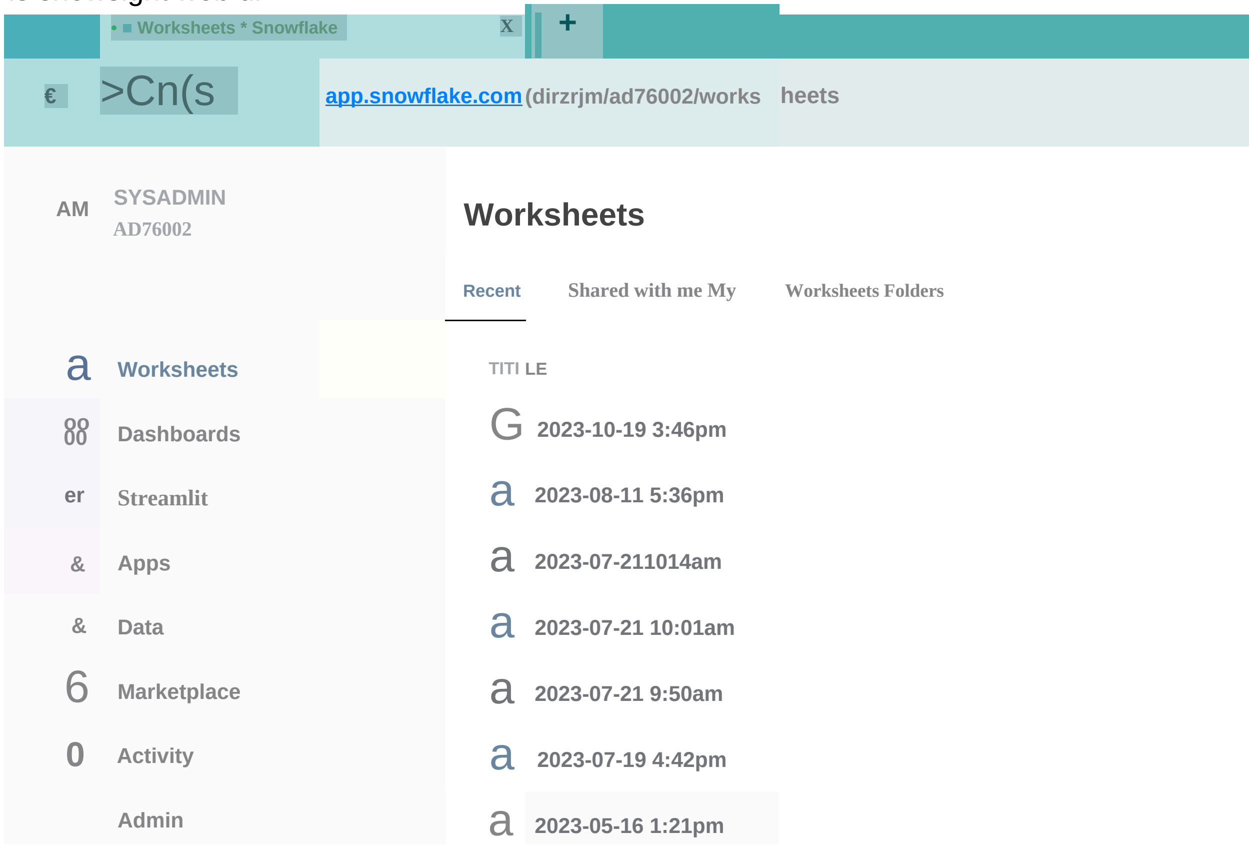
Tools and Interfaces

Snowflake has several tools available as part of the service.

- ***Classic Web UI—k*** browser-based web interface allowing users to work with objects and data in Snowflake. This was the original web interface that came out with Snowflake.

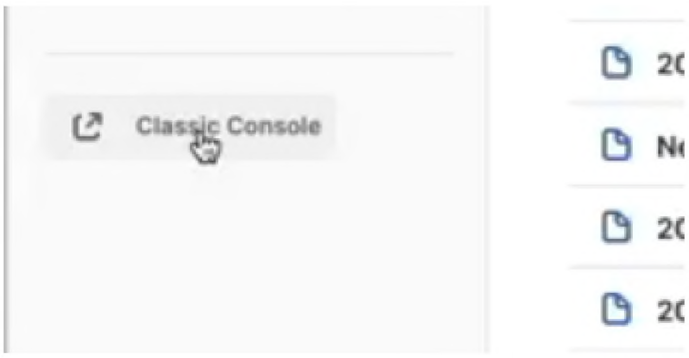
Snowsight—This is positioned as the next generation web interface, the successor to the Classic Web UI if you like. It contains all the features of the Classic Web UI but packaged in a different way. It also comes with a range of new capabilities, such as the ability to visualize data using graphs and charts. Snowsight allows you to create dashboards and filters and share them with other users, all within the web interface. It won't replace a dedicated data visualization tool, but it will satisfy the need for efficient analysis of data without needing to move the data out of Snowflake.

Snowsight is the newest UI for Snowflake
Default is snowsight web ui

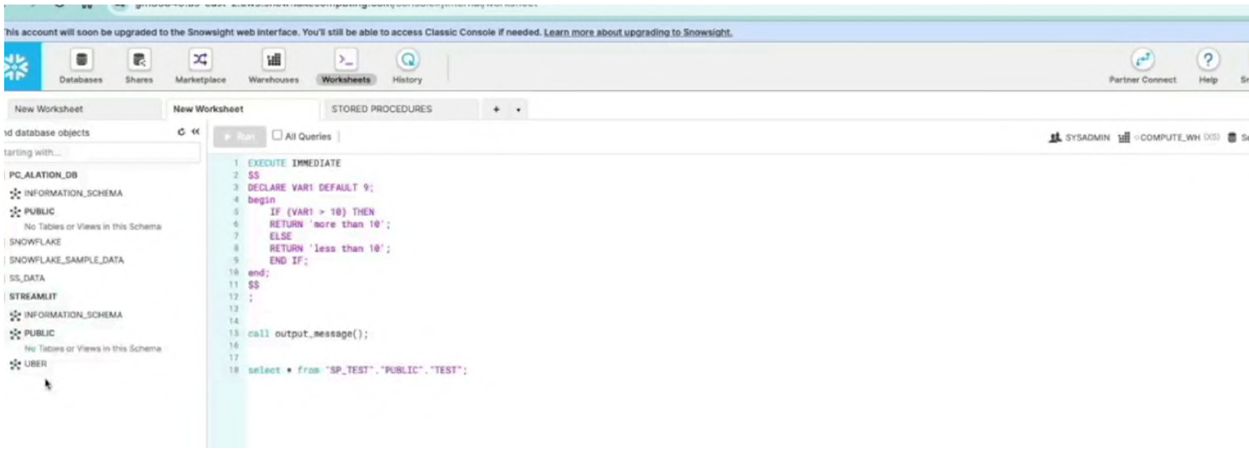


2023-10-19 3:46pm
2023-08-11 5:36pm
2023-07-211014am
2023-07-21 10:01am
2023-07-21 9:50am
2023-07-19 4:42pm
2023-05-16 1:21pm

To go back to classic console

- 

classic ui below



This account will soon be upgraded to the Snowsight web interface. You'll still be able to access Classic Console if needed. Learn more about upgrading to Snowsight.

```
PC_ALATION_DB
  INFORMATION_SCHEMA
  PUBLIC
    No Tables or Views in this Schema
SNOWFLAKE
SNOWFLAKE_SAMPLE_DATA
SS_DATA
STREAMLIT
  INFORMATION_SCHEMA
  PUBLIC
    No Tables or Views in this Schema
  UBER
```

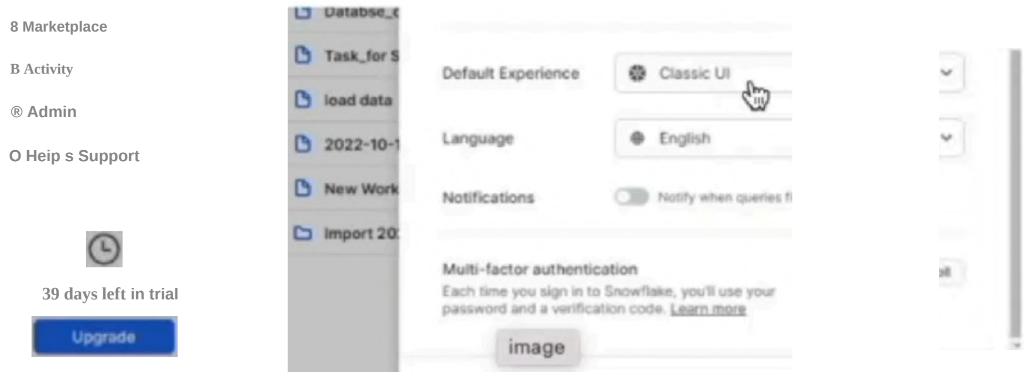
```
EXECUTE IMMEDIATE
$$
DECLARE VAR1 DEFAULT 9;
begin
    IF (VAR1 > 10) THEN
    RETURN 'more than 10';
    ELSE
    RETURN 'less than 10';
    END IF;
end;
$$
;


call output_message();


select * from "SP_TEST"."PUBLIC"."TEST";
```

Default experience

@ **Rupalil8**

a year ago

i have change the setting as per your instruction..but still it is open default view.



Default Experience: Classic UI
Language: English
Notifications: Notify when queries fi

Multi-factor authentication
Each time you sign in to Snowflake, you'll use your password and a verification code. Learn more

ak ____

Expand Post

SnowSQL is command line interface

- Used for bash

Tools and Interfaces

- ***SnowSQL*—**This is the command line interface (CLI) for Snowflake. It allows users to execute SQL queries, including unloading out of and loading data into Snowflake. You need to download and install this on your own machine to use it.

- ***Snowpark*—**This is a recent addition and a rapidly evolving area of Snowflake. *Snowpark* is aimed at data engineers and data scientists who wish to interact with querying and processing data in a data pipeline on Snowflake. It allows users to interact with the data using Scala or Java, with support for Python coming very soon. It provides the ability to write custom code and push this logic down to where the data lives in Snowflake.

— •omemrcmazn

A A

Streamlit

Tools and Interfaces

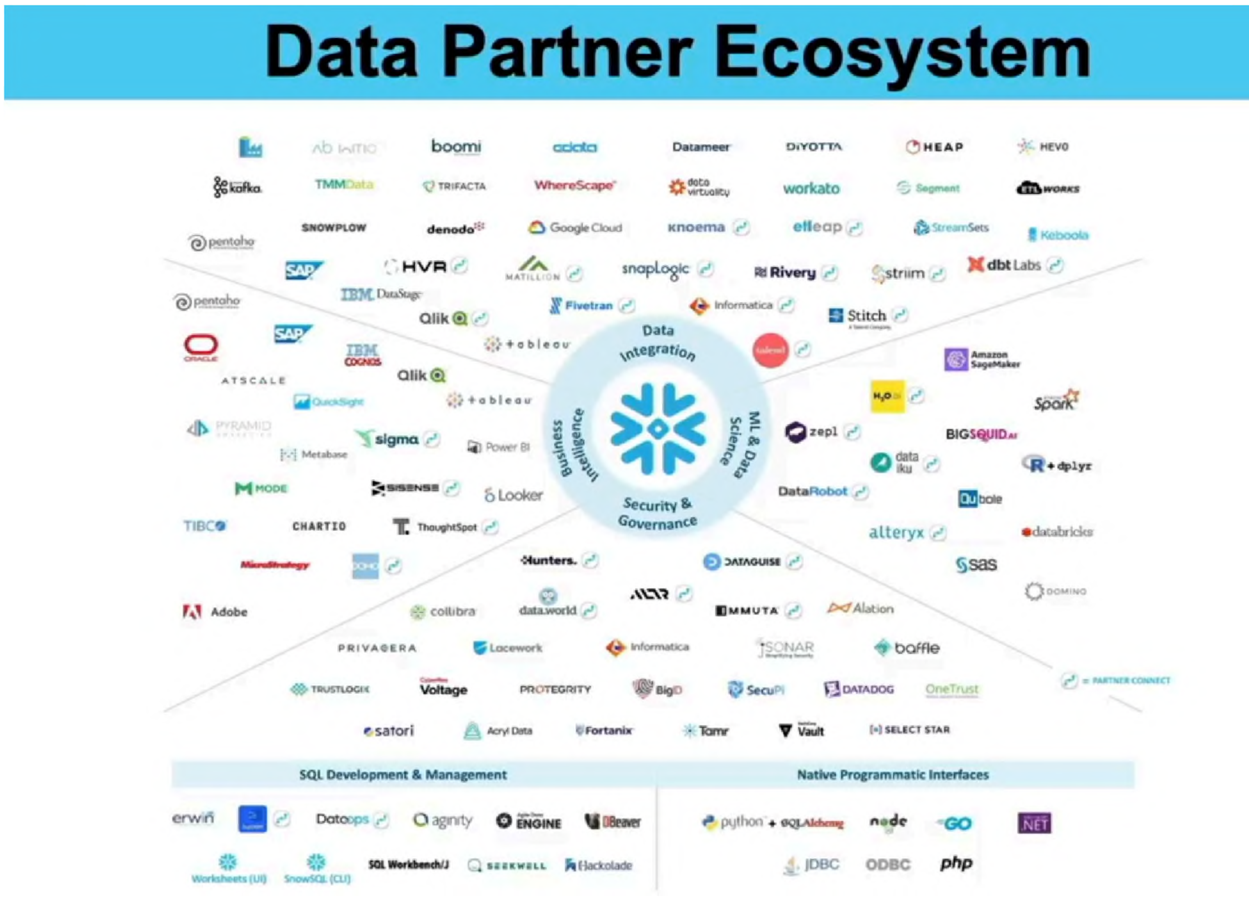
- ***Streamlit*** —Streamlit is an open-source Python library that makes it easy to create and share custom web apps for machine learning and data science. By using Streamlit you can quickly build and deploy powerful data applications. For more information about the open-source library, see the Streamlit Library documentation.

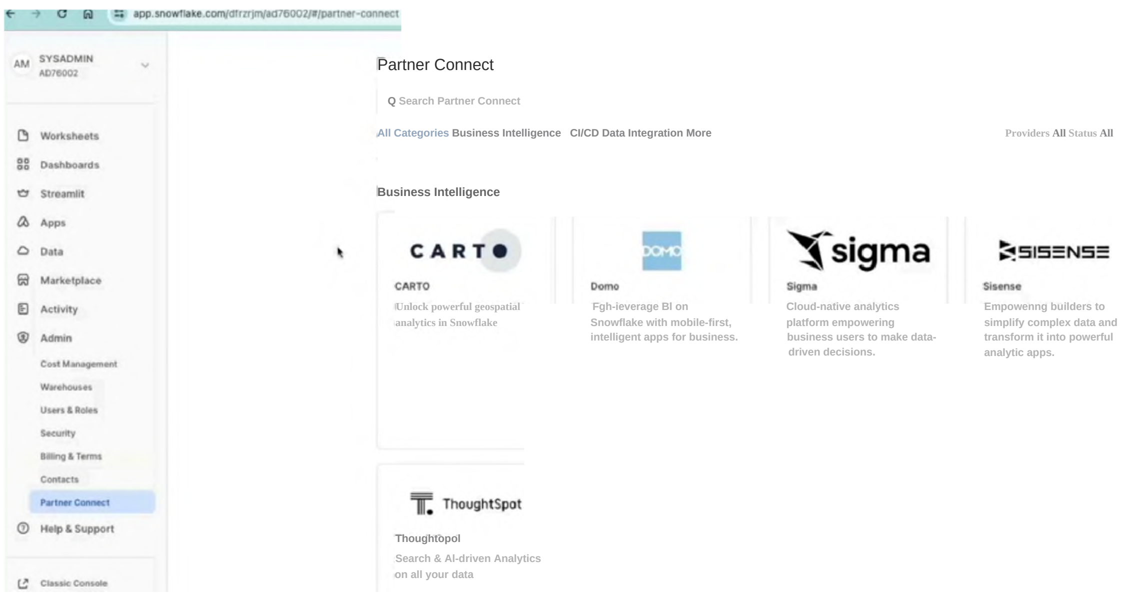
Streamlit in Snowflake helps developers securely build, deploy, and share Streamlit apps on Snowflake's data cloud. Using Streamlit in Snowflake, you can build applications that process and use data in Snowflake without moving data or application code to an external system.

A A

streamlit.io

Partner Ecosystem




app.snowflake.com/dfrzrjm/ad76002/#/partner-connect

SYSADMIN
AD76002

Worksheets
Dashboards
Streamlit
Apps
Data
Marketplace
Activity
Admin
Cost Management
Warehouses
Users & Roles
Security
Billing & Terms
Contacts
Partner Connect
Help & Support
Classic Console

Partner Connect

Search Partner Connect

All Categories Business Intelligence CI/CD Data Integration More

Providers All Status All

Business Intelligence

CARTO
Unlock powerful geospatial analytics in Snowflake

Domo
Fgh-ieverage BI on Snowflake with mobile-first, intelligent apps for business.

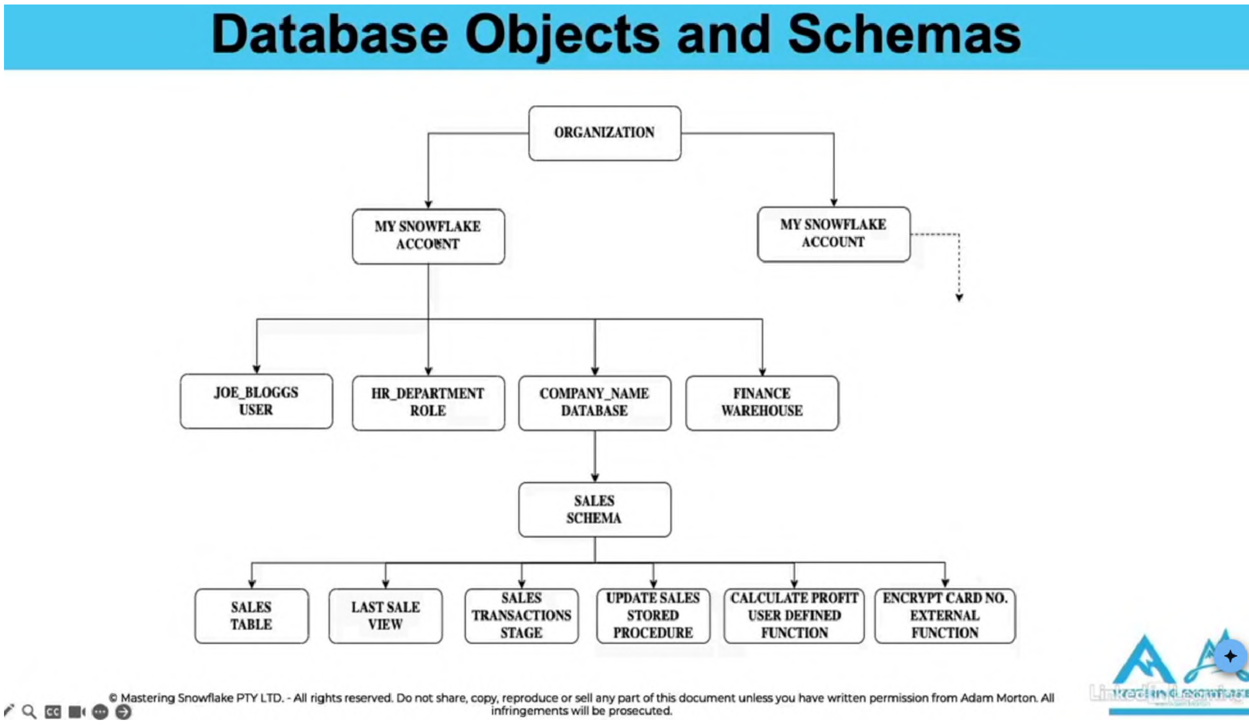
Sigma
Cloud-native analytics platform empowering business users to make data-driven decisions.

Sisense
Empowenng builders to simplify complex data and transform it into powerful analytic apps.

ThoughtSpot
Thoughtspol
Search & Al-driven Analytics on all your data

partner tools come integrated in snowflake

Database Objects and Schemas



- Table Types

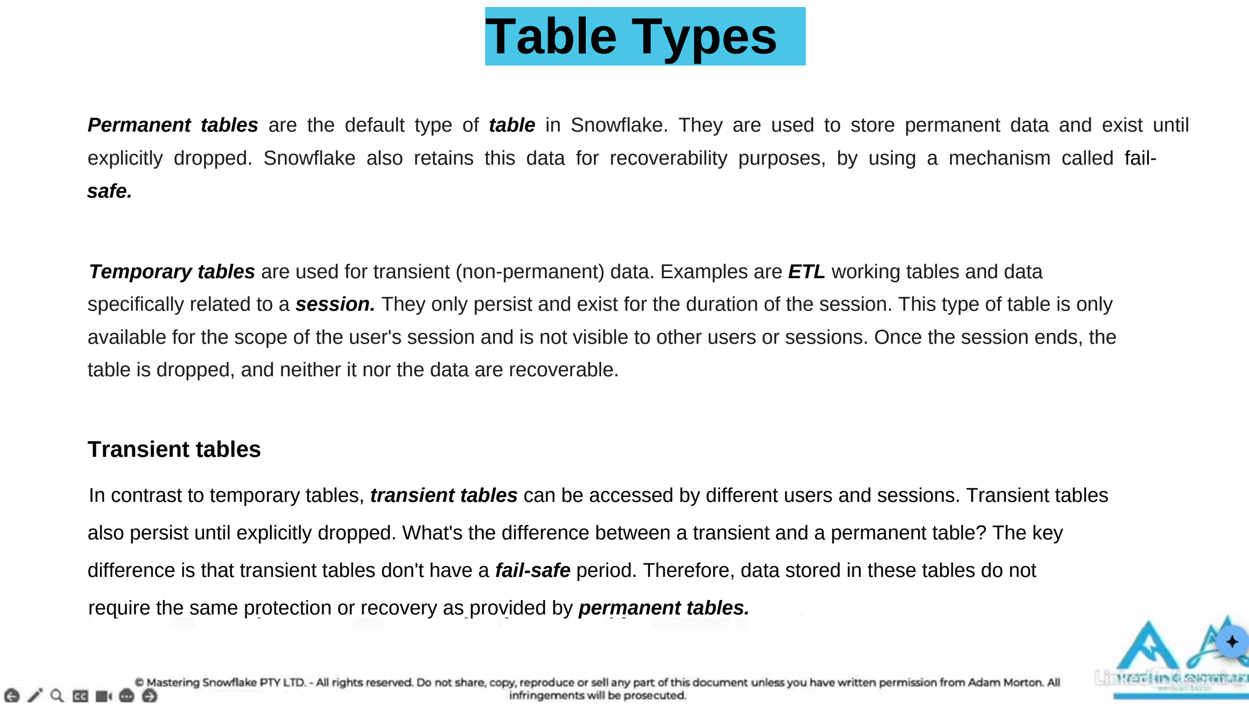


Table Types

Permanent tables are the default type of ***table*** in Snowflake. They are used to store permanent data and exist until explicitly dropped. Snowflake also retains this data for recoverability purposes, by using a mechanism called ***fail-safe.***

Temporary tables are used for transient (non-permanent) data. Examples are ***ETL*** working tables and data specifically related to a ***session.*** They only persist and exist for the duration of the session. This type of table is only available for the scope of the user's session and is not visible to other users or sessions. Once the session ends, the table is dropped, and neither it nor the data are recoverable.

Transient tables

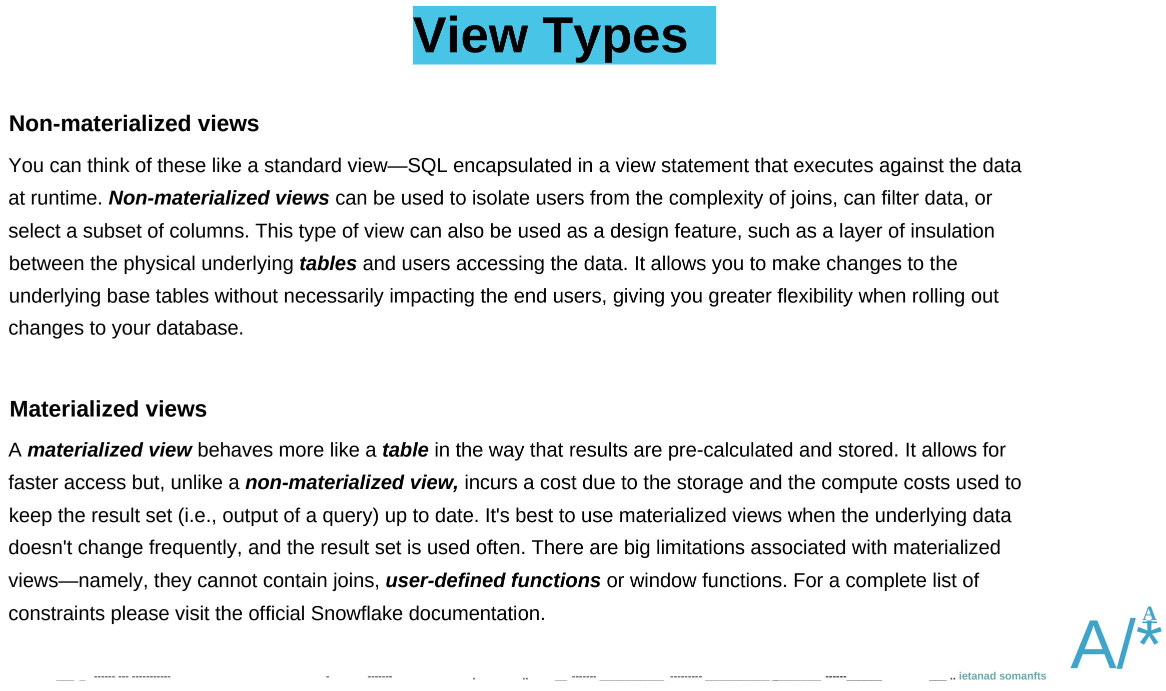
In contrast to temporary tables, ***transient tables*** can be accessed by different users and sessions. Transient tables also persist until explicitly dropped. What's the difference between a transient and a permanent table? The key difference is that transient tables don't have a ***fail-safe*** period. Therefore, data stored in these tables do not require the same protection or recovery as provided by ***permanent tables.***

Permanent, temporary, and transient[3]
(actually 9 tables)

[3] Table types

Permanent vs transient – transient does not have a fail-sale period; no protection or recovery as with permanent

View Types



View Types

Non-materialized views

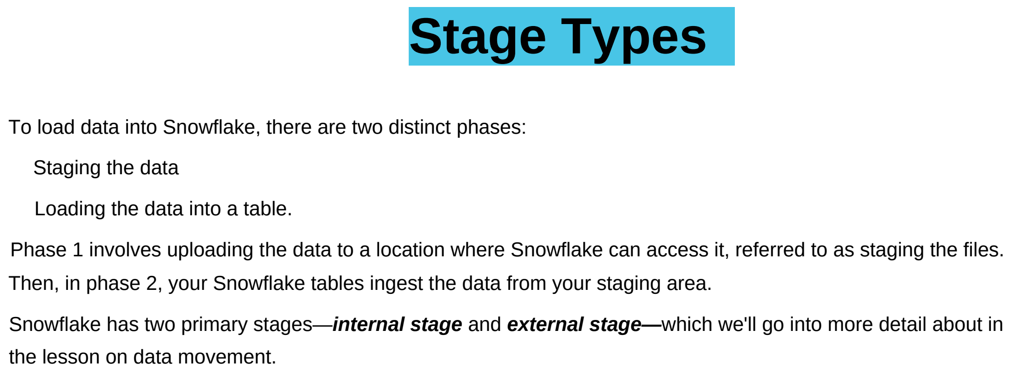
You can think of these like a standard view—SQL encapsulated in a view statement that executes against the data at runtime. ***Non-materialized views*** can be used to isolate users from the complexity of joins, can filter data, or select a subset of columns. This type of view can also be used as a design feature, such as a layer of insulation between the physical underlying ***tables*** and users accessing the data. It allows you to make changes to the underlying base tables without necessarily impacting the end users, giving you greater flexibility when rolling out changes to your database.

Materialized views

A ***materialized view*** behaves more like a ***table*** in the way that results are pre-calculated and stored. It allows for faster access but, unlike a ***non-materialized view,*** incurs a cost due to the storage and the compute costs used to keep the result set (i.e., output of a query) up to date. It's best to use materialized views when the underlying data doesn't change frequently, and the result set is used often. There are big limitations associated with materialized views—namely, they cannot contain joins, ***user-defined functions*** or window functions. For a complete list of constraints please visit the official Snowflake documentation.

SECURE VIEW prevents users from seeing underlying SQL

Stage Types



Stage Types

To load data into Snowflake, there are two distinct phases:

Staging the data

Loading the data into a table.

Phase 1 involves uploading the data to a location where Snowflake can access it, referred to as staging the files. Then, in phase 2, your Snowflake tables ingest the data from your staging area.

Snowflake has two primary stages—***internal stage*** and ***external stage—***which we'll go into more detail about in the lesson on data movement.

AWS - S3, Google Cloud, Microsoft - Azure

External Functions

External Functions

You can use ***external functions*** in Snowflake when you need to access external API services, such as machine learning models or geocoding functionality. You can pass data from Snowflake to an external function, obtain a result set, and write this back to Snowflake. You could also use external functions to 'push' data into another application.

Stored Procedures

Stored Procedures

A ***stored procedure*** allows you to encapsulate business logic to promote code re-use and make it easier and more efficient to manage your code base. In Snowflake, you can write a stored procedure in any of the following languages:

JavaScript

Snowflake Scripting (using SQL)

Scala (using Snowpark)

Java (using Snowpark).

Within the ***stored procedure*** logic, you can also cater for error handling, dynamically build up a SQL statement before executing it, and allow for branching and looping—none of which standard SQL supports.

Data Types

Data Types

Snowflake supports most basic SQL ***data types.*** You can convert data types to another type—for example, from a float to an integer. Depending on the conversion you are carrying out, you could lose information, so careful testing is required. Some data types are converted implicitly; others require explicit casting.

The certification exam won't go into detail about the SQL data types used for relational data, but it is worth familiarizing yourself with them. See the following link for detail about Snowflake's data types: https://docs.snowflake.com/en/sql-reference/intro-summary-data-tvpes.html

Geospatial Data

Geospatial Data

A relatively new data type in Snowflake—which may come up in the exam—is the geography data type. It allows you to work with geospatial data using degrees of longitude and latitude. A direct line between two points on a map is calculated while considering the curvature of the Earth's surface.

If you are working with geospatial data of this nature, make sure you store it within a geography data type and not in a varchar, variant, or number column. This will ensure you get the maximum possible performance when performing any geospatial calculations.

Store geography in this data type

Non-relational Data

Variant data type

Non-Relational Data

In addition to the basic SQL types (and the geography data type), Snowflake supports storing ***semi-structured data*** and querying ***unstructured data.***

This means you can load non-relational data directly into a table. Snowflake then provides extensions to the SQL language so that you can extract and work with the data by using SQL queries.

Working with ***semi-structured data*** in this way requires a special kind of data type called a ***variant.***

Understanding how to work with ***semi-structured*** data is not only important for the certification exam—it will benefit you when you come across semi-structured data in the field.

Row-Optimized Storage

Row *1*	1
	Joe Bloggs
	London
	100
	Jon
Row 2	*2*
	Anne Frozer
	Plymouth
	125
	Jan
Row 3	*3*
	William Mealing
	Gloucester
	200
	Jan

OLTP - online transactional processing
OLAP - online analytical processing
Data stored contiguously in columns

Column Orientated Databases

Column Orientated Databases

Order ID	1
	2
	3
	4
	5
Customer Name	Joe Bloggs
	Anne Frazer
	William Mealing
	Jim Hillier
	Andrew Sweeney
	Barrie James
City	London
	Plymouth
	Gloucester
	Cardiff
	Glasgow
	Newcastle

Compression is achieve more efficiently in column databases

Micro-Partitions

Micro-Partitions

Order ID	1	2	3
Customer Name	Joe Bloggs	Anne Frazer	William Mealing
City	London	Plymouth	Gloucester
Amount	100	125	200
Month	Jan	Jan	Jan

Order ID	4	5	6
Customer Name	Jim Hillier	Andrew Sweeney	Barrie James
City	Cardiff	Glasgow	Newcastle
Amount	90	133	110
Month	Feb	Feb	Feb

Micro-Partitions

Micro-partitions are created while loading data onto Snowflake. When you load data onto Snowflake, a few activities happen behind the scenes that are completely invisible to you as the user. You can break these operations down roughly as follows:

1. Divide and map the incoming data into micro-partitions using the ordering of the data as it is inserted/loaded.
2. Compress and encrypt the data.
3. Capture and store metadata.

Query Pruning
Snowflake targets partitions relevant only to the query, reduces credits

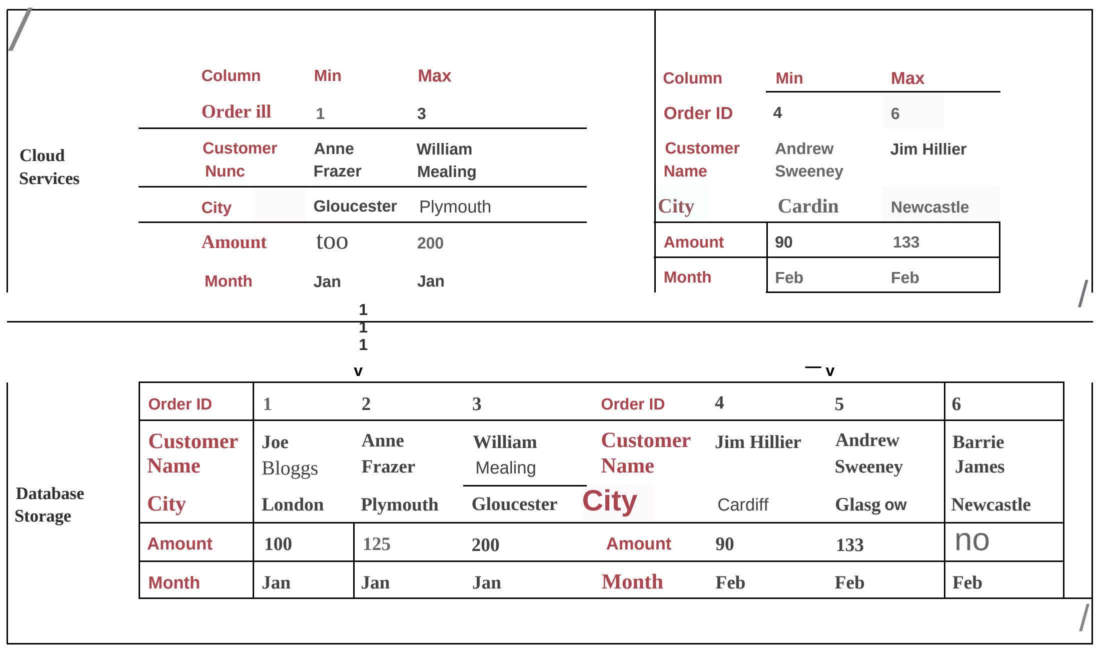
Query Pruning

Query pruning can be broken down into two distinct stages:

1. When an SQL query is executed with a WHERE clause, the metadata is used to locate those micro-partitions that hold the required data. So instead of searching through all micro-partitions, Snowflake targets just those that are relevant. This approach significantly reduces the execution time, as only the relevant data in the cloud storage layer is scanned.

2. Once the relevant micro-partitions have been identified in phase one, the second phase of query pruning is applied. The header of each partition is read to identify the relevant columns, further negating the need to read more data than is required.

Query Pruning



Column	Min	Max
Order ill	1	3
Customer Nunc	Anne Frazer	William Mealing
City	Gloucester	Plymouth
Amount	too	200
Month	Jan	Jan

Column	Min	Max
Order ID	4	6
Customer Name	Andrew Sweeney	Jim Hillier
City	Cardin	Newcastle
Amount	90	133
Month	Feb	Feb

Order ID	1	2	3	Order ID	4	5	6
Customer Name	Joe Bloggs	Anne Frazer	William Mealing	Customer Name	Jim Hillier	Andrew Sweeney	Barrie James
City	London	Plymouth	Gloucester	City	Cardiff	Glasg ow	Newcastle
Amount	100	125	200	Amount	90	133	no
Month	Jan	Jan	Jan	Month	Feb	Feb	Feb

Data Clustering

Data Clustering

Data in tables is clustered using a clustering key, which is a subset of columns in a table. The clustering key aims to co-locate data in the same micro-partitions. A clustering key can be applied to a table or a materialized view. If no clustering key is specified, data is clustered along natural dimensions such as date—which is typically the order that data is loaded.

Alternatively, you can choose to define a clustering key when creating a table, as the following code illustrates:

```
CREATE TABLE Sales
CLUSTER BY (Order_Id);
```

Generally, there's no need to define a clustering key for tables less than 1 TB. If you find yourself in this situation and query performance is suffering

Storage Layer

When you load and subsequently store data in Snowflake, the data physically resides on the underlying cloud platform provider's storage layer. Once the data is inside Snowflake, keep in mind that you'll need to use Snowflake credits to work with the data because you cannot access this data outside of Snowflake.

While the price does vary across different geographical regions and storage classes, AWS charges around USD $26 per TB per month to store data in Snowflake. Snowflake will simply pass on these costs to the user as the customer. The storage cost is calculated based on an average TB figure for the previous billing month.

Domain 2.0: Account Access and Security

Managing Snowflake accounts

Managing Snowflake Accounts

The Snowflake database is a read-only, shared database that exists in every Snowflake account. Its purpose is to store metadata along with historical usage metrics concerning the objects in your ***organization*** and accounts.

When Snowflake provides you with an account, the SNOWFLAKE database is imported from a ***share*** named ACCOUNT_USAGE. This database contains the ***schemas*** described in the table on the next slide.

A M

Snowflake database schemas

Snowflake database schemas

Schema Name	Description
ACCOUNT_USAGE	A set of ***views*** that displays object metadata for your account.
CORE	This ***schema*** stores data related to some Snowflake features. Currently, only system tags, which are part of the data classification feature, are stored here. This schema will grow over time as new features are released.
DATA_SHARING_USAGE	Contains metadata relating to ***data sharing,*** the ***data marketplace,*** or ***data exchange.***
INFORMATION_SCHEMA	This ***schema*** is automatically created in every single database within your account. We'll cover this in more detail later in this section.
ORGANIZATION _USAGE	This contains historical usage metadata across accounts in your organization.
READ_ACCOUNT_USAGE	This schema is like ACCOUNT_USAGE but with one key difference—it is focused on ***reader accounts*** if you have any setup within your environment.

Account Usage

Account Usage

This schema contains a mixture of views and table functions for ***database objects*** as well as information on ***roles, warehouses,*** and databases.

Account usage views

The account usage views in Snowflake contain information relating to the entire Snowflake account, and may include historical information. At the time of writing, there are nearly 50 different views available. For a full list, head to the official Snowflake documentation here: https://docs.snowflake.com/en/sql-reference/account-usage.html#account-usage-views. In this section, we'll just touch on the views that may come up in the exam.

A ***A***

Account usage

Access history

The access history view contains data relating to whenever a user executes a query to read data or when the SQL statement performs a write operation, such as INSERT, UPDATE, and DELETE, along with variations of the COPY command, from the source to the target data object.

Each row in the access history view contains a single record per SQL statement. The record describes the columns the query accessed directly and indirectly (i.e., the underlying tables that the data for the query comes from). These records facilitate regulatory compliance auditing and provide insights into popular and frequently accessed tables and columns since there is a direct link between the user (i.e., query operator), the query, the table or view, the column, and the data.

Account_Usage vs Information Schema

This schema contains a mixture of views and table functions for ***database objects*** as well as information on ***roles, warehouses,*** and databases. This table highlights the key differences:

Difference	Account Usage	Information Schema
Includes dropped objects	**Yes**	**No**
Latency of data	**Between 45 mins to 3 hours**	**Zero latency**
Retention of historical data	**1 year**	**7 days to 6 months, depending on the view or table function**

Understanding Grants

GRANTS_TO_ROLE

This view allows you to understand the ***access control*** privileges granted to a role.

GRANTS_TO_ USERS

This view allows you to understand the ***access control*** privileges granted to an individual user.

We cover more information on granting, viewing, and revoking privileges in the section on access control later in this section.

Account Identifiers

An ***account identifier is*** used when you need to connect to a Snowflake account. If your organization has multiple Snowflake instances, an account identifier will help you distinguish between them.

You'll use an account identifier when connecting to Snowflake using any of the following methods:

1. Accessing the ***Classic Web UI*** or ***Snowsight***
2. ***SnowSQL*** or other connectors and drivers—e.g., ODBC, JDBC
3. Overarching global features such as secure **data sharing** and ***database replication.***

Using a URL to connect to Snowflake, for example, the ***account identifier*** appears in the URL as illustrated below:

<account_identifer>.snowflakecomputing.com

For example, your URL might look like this, where gm3456 is your ***account identifier:***

gm3456.snowflakecomputing.com

Snowflake Organization

Snowflake Organisation

An ***organization*** is a Snowflake object that links the accounts owned by your business entity; it comes into play when you have more than one Snowflake account. For example, you might have multiple Snowflake accounts when data needs to be replicated on different cloud providers to support disaster recovery scenarios.

Let's say your primary Snowflake account runs on AWS in the US West (Oregon) region, and that region goes offline. You could failover to a backup Snowflake account that has a copy of your data running on, for example, Microsoft Azure (West US 2). In this case, an ***organization*** ties both these accounts together under the same umbrella.

If you have an ***organization,*** your ***organization name*** is prefixed to your ***account name*** to create your ***account identifier.***

Managing Costs

Managing Costs

Snowflake costs have three categories—compute, data storage, and data transfer. To get visibility of the costs associated with your Snowflake account, Snowflake has some built-in reporting within the ***web ui.***

These options might not be flexible enough for you, or you might need to generate custom reporting on the spend to share it with senior managers, or those people who don't need access to Snowflake. In that case, you can use the metadata in the ***information schema—***or the account usage views—which we covered earlier in this chapter.

Organizations also allow you to roll up the spend across multiple Snowflake accounts at the overall level. In ***Snowsight,*** you can go to the Admin > Accounts section to ***view*** this information.

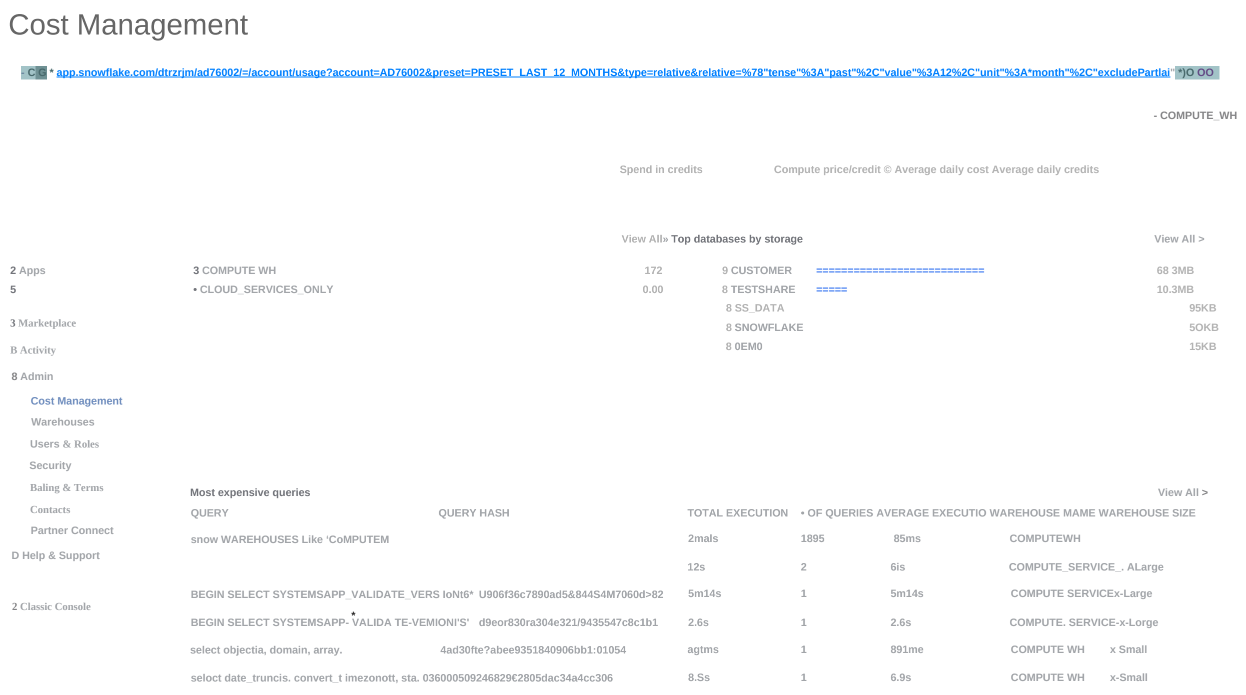
Resource Monitors

Resource Monitors

The last thing you want to get is an unexpectedly huge bill at the end of the month for your Snowflake account. And believe me, it does happen!

Of the three categories you pay for, the compute resources associated with ***virtual warehouses*** are typically the largest proportion. ***Resource monitors*** are attached to user-defined virtual warehouses (not managed compute provided by Snowflake!) and monitor the credit usage of the virtual warehouse.

Resource monitors can only be created by the ACCOUNTADMIN role and can be configured in the web interface ***(Classic Web UI*** or ***Snowsight)*** or by using SQL. Limits can be set for a specified interval or date range. When these limits are reached, or you are getting close to a pre-defined threshold, the ***resource monitor*** can trigger various actions, such as sending alert notifications or suspending user-managed virtual warehouses.



Cost Management

app.snowflake.com/dtrzrjm/ad76002/=/account/usage?account=AD76002&preset=PRESET_LAST_12_MONTHS&type=relative&relative=%78"tense"%3A"past"%2C"value"%3A12%2C"unit"%3A*month"%2C"excludePartlai"

- COMPUTE_WH

Spend in credits

Compute price/credit © Average daily cost Average daily credits

View All» Top databases by storage

View All >

2 Apps
5

3 Marketplace

B Activity

8 Admin

Cost Management
Warehouses
Users & Roles
Security
Baling & Terms
Contacts
Partner Connect

D Help & Support

2 Classic Console

3 COMPUTE WH	172
• CLOUD_SERVICES_ONLY	0.00

9 CUSTOMER	68 3MB
8 TESTSHARE	10.3MB
8 SS_DATA	95KB
8 SNOWFLAKE	5OKB
8 0EM0	15KB

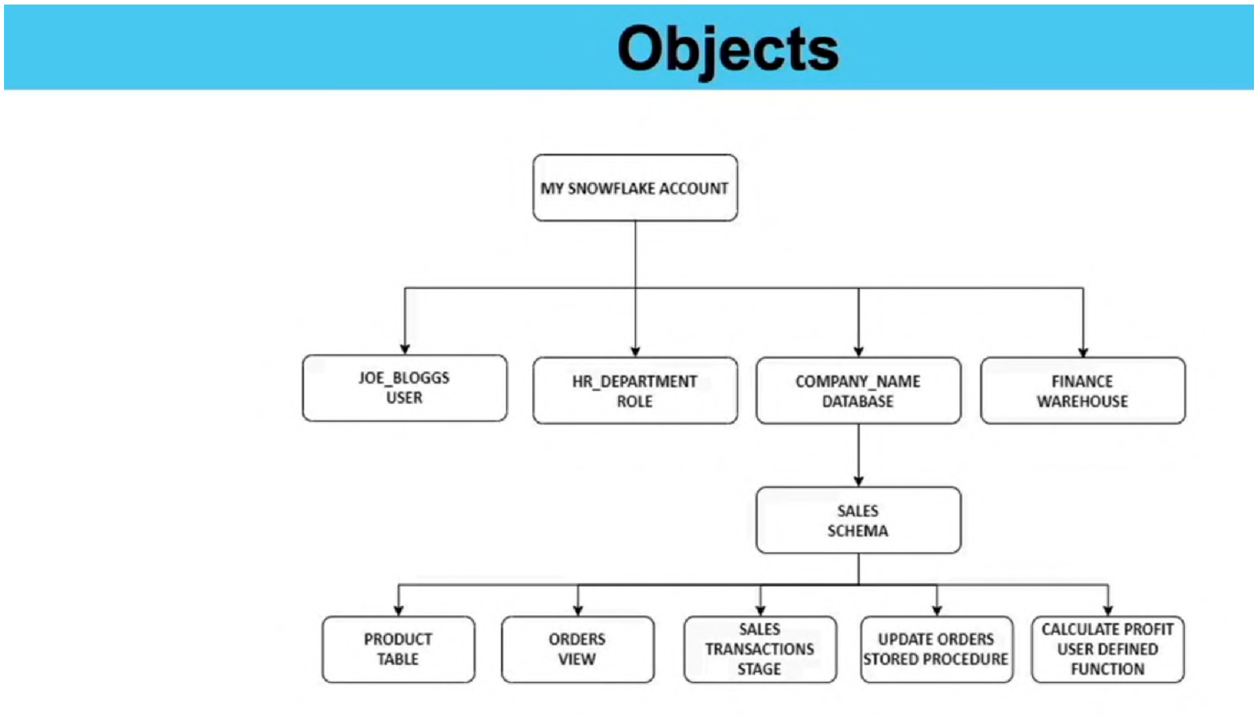
Most expensive queries

View All >

QUERY	QUERY HASH	TOTAL EXECUTION	• OF QUERIES	AVERAGE EXECUTIO	WAREHOUSE MAME	WAREHOUSE SIZE
snow WAREHOUSES Like 'CoMPUTEM		2mals	1895	85ms	COMPUTEWH	
		12s	2	6is	COMPUTE_SERVICE_. ALarge	
BEGIN SELECT SYSTEMSAPP_VALIDATE_VERS IoNt6*	U906f36c7890ad5&844S4M7060d>82	5m14s	1	5m14s	COMPUTE SERVICEx-Large	
BEGIN SELECT SYSTEMSAPP- VALIDA TE-VEMIONI'S'	d9eor830ra304e321/9435547c8c1b1	2.6s	1	2.6s	COMPUTE. SERVICE-x-Lorge	
select objectia, domain, array.	4ad30fte?abee9351840906bb1:01054	agtms	1	891me	COMPUTE WH	x Small
seloct date_truncis. convert_t imezonott, sta.	036000509246829€2805dac34a4cc306	8.Ss	1	6.9s	COMPUTE WH	x-Small

Access Control

Objects



Roles

Role	Description
ACCOUNTADMIN	The most powerful role available, it administers the Snowflake account and can view all credit and billing information. The other admin roles are children of this role: SYSADMIN and SECURITYADMIN. Snowflake recommends limiting the use of this account and restricting access to a minimum set of users. You should avoid setting this role as a default for any users. Additionally, I would strongly recommend configuring multi-factor authentication (MFA), which we'll cover later in this chapter, on this account.
SYSADMIN	This role can create warehouses and databases and all objects within a database ***(schemas, tables, views,*** etc.).
SECURITYADMIN	This role is designed to cater to the administration of security. It includes the management of granting or revoking ***privileges*** to ***roles.***
USERADMIN	This role is used for creating roles and users and managing the privileges assigned to them. The USERADMIN role is a child of the SECURITYADMIN role.
PUBLIC	The PUBLIC role is a default role all users end up in automatically. It provides privileges to log in to Snowflake and some basic object access.

Access control privileges

Privileges are granted to ***roles,*** and roles are granted to users, to specify the operations that the users can perform on objects in the system. Before we get into granting privileges, let's touch on what privileges we'll use in this example and where you can go for the complete list.

For a full list of all ***access control*** privileges, please review the official Snowflake documentation here: https://docs.snowflake.com/en/user-guide/security-access-control-privileges.html

Virtual warehouse privileges

Privilege	Usage
MODIFY	Provides the ability to change the size and configuration settings of the ***virtual warehouse.***
OPERATE	Allows the ***role*** to start, stop and suspend a ***virtual warehouse.***
USAGE	Allows the ***role*** to use the available compute resources from the ***virtual warehouse.***
OWNERSHIP	Grants full control over a ***warehouse.*** Only a single ***role*** can hold this ***privilege*** on a specific object at a time.
MONITOR	Enables viewing current and past queries executed on a ***warehouse*** as well as usage statistics on that warehouse.

Examples of privileges

Schema privileges (subset)

Privilege	Usage
MODIFY	Enables altering any settings of a ***schema.***
USAGE	Enables using a ***schema.***
CREATE TABLE	Enables creating a new table in a ***schema,*** including ***cloning*** a table.

Role Hierarchy

Role Hierarchy

A ***role hierarchy*** sounds complex, doesn't it? Well, in fact, once you get your head around the concept of granting and revoking privileges, it's quite straightforward.

You can start creating a custom role hierarchy by executing the following SQL statement:

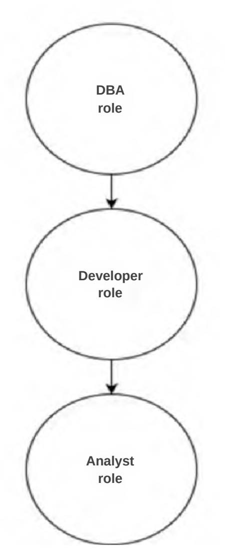
```
USE ROLE SECURITYADMIN;
CREATE ROLE DBA;
CREATE ROLE Analyst;
CREATE ROLE Developer;
```

Privilege inheritance

The ***privileges*** of the role then cascade down the ***role hierarchy.***

It's much easier to explain by continuing our example:

```
GRANT ROLE Analyst TO ROLE Developer;
GRANT ROLE Developer TO ROLE DBA;
```



Privilege inheritance

At this stage, we have created three new roles and linked them to form a hierarchy. However, we haven't assigned any privileges yet, so the roles cannot access anything. Let's start opening access by applying privileges to the roles.

For our analyst role, we want to give them read-only access, so they'll need the following grants:

USAGE on a virtual warehouse for compute resources

USAGE on the sales database

SELECT on all tables

Granting Privileges

```
GRANT USAGE ON WAREHOUSE main_wh TO role analyst;
GRANT USAGE ON DATABASE sales TO role analyst;
GRANT USAGE ON SCHEMA sales.staging TO role analyst;
```

Granting Privileges

We want our developer role to have the following:

- USAGE on a virtual warehouse for compute resources (inherited from Analyst)
- USAGE on the datasets for data pipelines and ingestion use cases (inherited from Analyst)
- CREATE VIEW and CREATE TABLE to create these objects as required.

```
GRANT CREATE TABLE, CREATE VIEW ON SCHEMA sales.staging TO role developer;
```

Granting Privileges

Finally, we come on to our DBA role. Remember that our DBA role has inherited all the privileges from both the Developer and Analyst roles. This means we only need to provide our DBA role with two final GRANTS:

- OPERATE on the virtual warehouse
- CREATE stages.

```
GRANT OPERATE ON WAREHOUSE main_wh TO ROLE DBA;
GRANT CREATE STAGE ON SCHEMA sales.staging TO ROLE DBA;
```

Privilege inheritance

```
USE ROLE USERADMIN;
CREATE USER user_1;
CREATE USER user_2;
CREATE USER user_3;

USE ROLE SECURITYADMIN;
GRANT ROLE DBA TO USER user_l;
GRANT ROLE developer TO USER user_2;
GRANT ROLE analyst TO USER user_3;
```

This completes setting up and configuring our ***role hierarchy.*** But what happens if we need to revoke some of these ***privileges?*** Let's look at how we do this.

Revoking Grants

Revoking Grants

Following Snowflake's ***role-based access control (RBAC)*** principle, we both assign (grant) and remove (revoke) privileges at the role level. Take a look at the following code:

```
REVOKE CREATE TABLE, CREATE VIEW ON SCHEMA sales.Staging
FROM role developer;
```

This statement is the same as the one we executed a little earlier in our example, with two key differences—we switched GRANT to REVOKE and changed the TO role to the FROM role.

Show Grants

Show Grants

The show grants command will list all access-control ***privileges*** that have been explicitly granted to ***roles,*** users, and ***shares.***

There are subtle variations in how you run this command depending on the information you require, as shown here.

Command	Variation	Description
SHOW GRANTS		The same as executing a query such as SHOW GRANTS TO USER <current_user>.
SHOW GRANTS ON	ACCOUNT <obiecttype> <obiectname>	ACCOUNT—Lists all the account-level (i.e., global) privileges granted to roles.
SHOW GRANTS TO	ROLE <role name> WUUUWWWVOUUUW USER <username> SHARE <sharename>	ROLE—Lists all privileges and roles granted to the role. USER—Lists all the roles granted to the user. SHARE—Lists all the privileges granted to the share.
SHOW GRANTS OF	ROLE <rolename> SHARE <sharename>	ROLE—Lists all users and roles to which the role has been granted. SHARE—Lists all the accounts for the share and indicates the accounts that are using the share.
SHOW FUTURE GRANTS IN	SCHEMA <database name ><schemaname> DATABASE <databasename>	SCHEMA—Lists all privileges on new (i.e., future) database objects in the schema granted to a role. DATABASE—Lists all privileges on new (i.e., future) objects of a specified type in the database granted to a role.

a AA------- c------ nu I rr AII --------- n. ---- ---

Future Grants

Future Grants

Future grants allow you to grant ***privileges*** ahead of time, saving you from having to re-apply the same privileges every time you add a new object to the database or ***schema.***

For our developer role, we want to add the following grant:

INSERT for adding records to new tables

```
GRANT INSERT ON FUTURE TABLES IN SCHEMA sales.staging TO role developer;
```

This ensures that for any new tables added to the staging schema, our developer role can still insert new records into these tables.

Multi-factor authentication (MFA)

MFA is not turned on by default. Users of Snowflake must self-enroll. They can do this by heading into their user preferences located in a dropdown menu next to the login name in both the ***Classic Web UI*** and ***Snowsight.***

It must be pointed out that MFA can be enforced for the following:

Classic Web UI

Snowsight

SnowSQL

JDBC

ODBC

Python.

AA

Data Encryption

Data Encryption

As data is loaded into Snowflake, it is encrypted at rest and in transit by default. This is always the case, and there is no option to turn this setting off. This method aims to secure data to prevent unauthorized third parties from reading it while it is at rest or in transit.

If you are using an ***external stage,*** you can choose to encrypt the data files. If you are using an ***internal stage*** to load local files into Snowflake, then Snowflake will first encrypt these files on your own local machine ***before*** it attempts to load them into the stage.

A A

Network Rules & Policies

It is possible to further secure access to Snowflake by adding a layer of security at the network level.

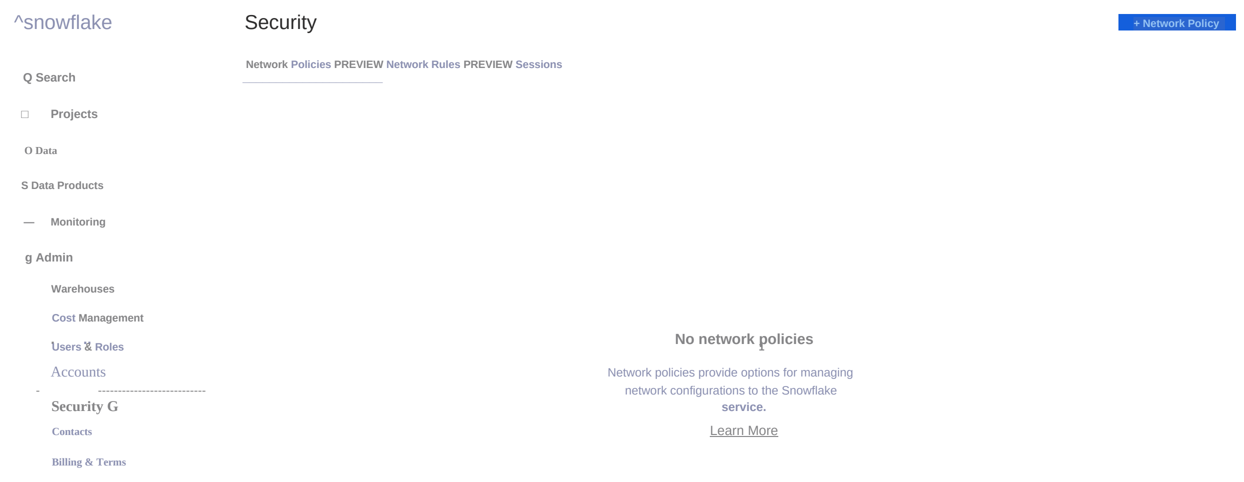
Network policies are provided by Snowflake to restrict access to your Snowflake account based on IP addresses. You can create a permitted list of IP addresses that can access your account as well as a blocked list of IP addresses.

Network Rules have been recently introduced to logically group together network identifiers. ***Network Rules*** are then referenced by ***Network policies*** to make it easier to manage your networking policies and rules.

Network rules & policies can be modified through ***Snowsight,*** the ***Classic Web UI,*** or SQL to add or remove network identifiers from the list of allowed and blocked addresses.

A AA

Can do this in Snowsight or SQL



Network policies provide options for managing network configurations to the Snowflake service.

Security

Network Policies PREVIEW Network Rules PREVIEW Sessions

	STATUS 4	NETWORK RULES	CREATED
MY_NETWORK_POLICY	Inactive	1 Allowed / 0 Blocked	4/16/2024, 8:40:34 PM

Federated authentication and Single sign-on

Federated authentication and Single sign-on (SSO)

In a federated environment, you effectively outsource the user authorization element from Snowflake to an external service. Typically, in a federated environment, a user would be authenticated with an external service maintained centrally, often referred to as an ***IdP*** or ***identity provider,*** such as Okta, to name a well-known ***IdP.***

(S adnmcsion

Data security

Row-level security

Row-level security

Row-level security is implemented using a feature called ***row-access policies,*** which allows you to secure data at the individual row level. You can instruct Snowflake to allow a user or application to see a particular row when querying the data based on the role executing the query.

You need to perform only three simple steps to use row-access policies to accomplish row-level security:

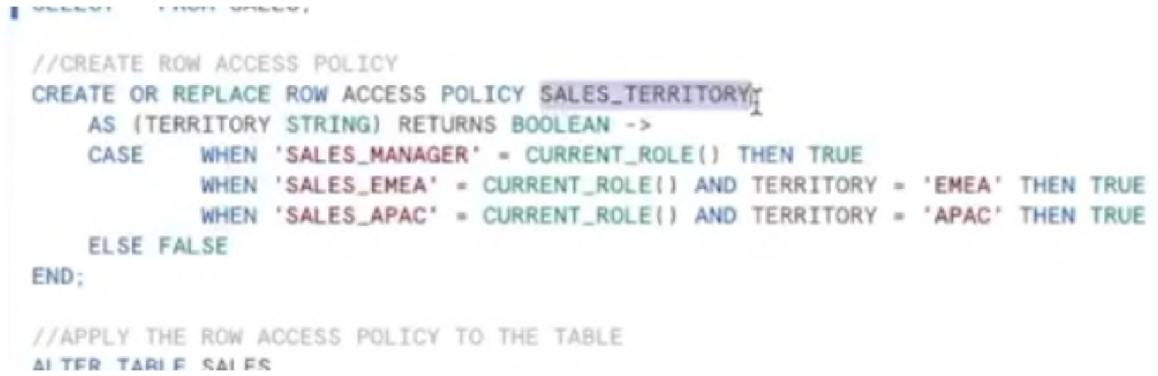
1. Define a policy and optionally define a mapping table.
2. Apply the policy to one or more tables.
3. Query the data.

Row-level security

A row-access policy contains an expression that can specify Snowflake ***database objects*** (e.g., ***table*** or ***view*** to determine which rows should be visible in a given context. Let's look at an example of what a policy might look like:

Access Policies

Below is creating an Access Policy



```
SELECT * FROM SALES;

//CREATE ROW ACCESS POLICY
CREATE OR REPLACE ROW ACCESS POLICY SALES_TERRITORY
    AS (TERRITORY STRING) RETURNS BOOLEAN ->
    CASE    WHEN 'SALES_MANAGER' = CURRENT_ROLE() THEN TRUE
            WHEN 'SALES_EMEA' = CURRENT_ROLE() AND TERRITORY = 'EMEA' THEN TRUE
            WHEN 'SALES_APAC' = CURRENT_ROLE() AND TERRITORY = 'APAC' THEN TRUE
    ELSE FALSE
END;

//APPLY THE ROW ACCESS POLICY TO THE TABLE
ALTER TABLE SALES
```

```
//APPLY THE ROW ACCESS POLICY TO THE TABLE
ALTER TABLE SALES
    I ADD ROW ACCESS POLICY SALES.TERRITORY
    | OH (TERRITORY);


    //CREATE ROLES
    USE ROLE SECURITYADMIN:
    CREATE OR REPLACE      ROLE  SALES.MANAGER;
    CREATE OR REPLACE      ROLE  SALES_EMEA:
    CREATE OR REPLACE      ROLE  SALES_APAC; r
                                                     1
    //GRANT PERMISSIONS ON OBJECTS TO ALL ROLES
    Pnaur • IC A ne nai wlDcuPE rnsnurrE tn l Tn Cas re ussiorn
```

```
•* l wnn'E VI nLTLPE "W- SRLte-mre

//GRANT PERMISSIONS ON OBJECTS TO ALL ROLES
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO SALES.MANAGER;
GRANT   USAGE    ON          WAREHOUSE COMPUTE _WH TO SALES_EMEA;
GRANT   USAGE    ON            WAREHOUSE COMPUTE_WH TO SALES.APAC;

GRANT   USAGE    ON           DATABASE ROW ACCESS TO SAL ES .MANAGE R.
GRANT   USAGE    ON             DATABASE ROW.ACCESS TO SALESJEMEA.
GRANT   USAGE    ON              DATABASE ROW.ACCESS TO SALES.APAC.

USE ROLE SYSADMIN:
GRANT USAGE ON SCHEMA PUBLIC TO SALES.MANAGER.
             ®      GRANT      USAGE ON SCHEMA    PUBLIC TO SALES.EMEA
GRANT   USAGE ON SCHEMA        PUBLIC TO SALES.APAC;

GRANT   SELECT ON TABLE         SALES TO SALES.MANAGER.
GRANT   SELECT ON TABLE         SALES TO SALES.EMEA;
GRANT   SELECT ON TABLE         SALES TO SALES.APAC;
```

> Results

```
//ADO THESE ROLES TO YOUR OWN USER TO MAKE IT EASY TO TEST OUT //FO
USE ROLE SECURITYADMIN;
GRANT ROLE SALES_MANAGER TO USER adammorton;
GRANT ROLE SALES_EMEA TO USER adammorton;
GRANT ROLE SALES_APAC TO USER adammorton;r

//TEST OUT THE DIFFERENT ROLES AND OBSERVE THE RESULTS
USE ROLE SALES.MANAGER;
SELECT TERRITORY, COUNT(*)
FROM SALES
GROUP BY TERRITORY;

USE ROLE SALESEMEA:
SELECT TERRITORY, COUNT(•)
FROM SALES
```

Results Chart

Mapping Tables

I uA I wn V _WLL TMvuru \ I t

```
INSERT INTO SALES.TERRITORY (ROLE. TERRITORY-CODE)
SELECT 'SALES.EMEA, 'EMEA'
UNION
SELECT 'SALES.APAC'. 'APAC';

| SELECT • FROM SALES.TERRITORY;

//CREATE ROW ACCESS POLICY

CREATE OR REPLACE ROW ACCESS POLICY SALES.TERRITORY
            AS (TERRITORY STRING) RETURNS BOOLEAN ->
 EXISTS (SELECT • FROM SALES.TERRITORY
          WHERE SALES.TERRITORY,TERRITORY-CODE -I TERRITORY
            AND SALES.TERRITORY ROLE • CURRENT.ROLE()p
```

Getting Info for Row-level security

Row-level security

Be aware that if you need to update an existing row-access policy, you might need to view its definition first. There are two ways of doing this:

1. Call the GET_DDL function.
2. Run the DESCRIBE ROW ACCESS POLICY command.

If you need to obtain a list of those objects that have a row-access policy applied to them, you can use the ***information schema*** POLICY_REFERENCES table function to do this. You can either return all objects attached to a specific policy, like so:

```
select *
from table(
  mydb.information_schema.policy_references(
    policy_name=>'<policy_name>'
```

Column-level security

Dynamic data masking uses ***masking policies,*** which obfuscate plain-text data in the table and view columns at execution time.

External tokenization allows ***data exchange*** for a ***token*** outside of Snowflake. The ***token*** is stored within Snowflake and cannot be read until it is ***detokenized.*** At runtime, if the user or application querying the ***tokenized*** data is permitted to see the underlying data, then ***masking policies*** and ***external functions*** are leveraged to ***detokenize*** the data.

Dynamic Data Masking

A masking policy is a schema-level ***database object*** to protect sensitive data from being viewed in plain text by unauthorized users. You can think of this as a layer of protection over the table. It acts as a gatekeeper and decides who can see the data in the column in question.

The data within the table is left untouched because the masking policy is applied at query execution time and acts like a filter between the user running the query and the actual data. This means you don't need to worry about applying complex transformations against the physical data to mask it.

Dynamic Data Masking

You can define a masking policy in different ways so that unauthorized users see masked, partially masked, obfuscated, or ***tokenized*** data.

Let's say you have customer email addresses stored in a column called CUSTOMER EMAIL. You have a requirement to prevent anyone other than the role called 'SENSITIVE_ALLOWED_ROLE' from viewing the email address.

External Tokenization

External tokenization ensures that it is still possible to group records by a tokenized value without exposing sensitive information—for example, group medical records by diagnosis code with the patient diagnosis code ***tokenized.***

Data analysts can then query a view on the diagnosis code to obtain a count of the number of patients with a unique diagnosis code. However, bear in mind that ***external functions*** cannot be included with the data-sharing functionality, while ***dynamic data masking*** can.

Object Tagging

Object Tagging

Tags enable data stewards to track sensitive data for a range of data-governance-related use cases.

You can assign a tag to a whole range of objects in your Snowflake account, from ***virtual warehouses*** to ***schemas,*** to ***tables, streams, views,*** columns, and more. You can think of it as a convenient way to categorize objects within your Snowflake environment.

Note: Tagging is a feature introduced at the Enterprise edition level.

Object Tagging

You define a tag once and attach it to one or more objects in your environment. Tags will also be inherited in line with the ***object hierarchy.*** So, applying the tag to objects higher in the object hierarchy results in the tag being applied to all child objects. For example, if a tag is set on a table, it will be inherited by all columns in that table. This concept is known as ***tag inheritance.***

Once tags have been assigned, you can 'discover' tags within your environment to track data that has been tagged as sensitive or find out which objects associated with a tag are consuming the most resources.

Secure Views

Secure views are a special kind of view to use in those situations where you don't wish to expose the underlying view definition to users. For security reasons, you may want to prevent unauthorized access to the underlying tables, or business logic, that make up the view definition.

When you create a view in Snowflake, it, by default, creates a standard, non-secure view. Adding the keyword 'SECURE' to the CREATE VIEW statement makes the view you've created secure.

You might be asking yourself the question—why not use ***secure views*** all the time? The reason is that the ***query optimizer*** bypasses some optimizations it can apply to standard, non-secure views for ***secure views.*** The result is a performance tradeoff; ***secure views*** do not perform as well as ***non-secure views.*** Snowflake's recommendation is to use them where you need to restrict access to the underlying data, not simply for query convenience.

A A

Domain 3.0: Performance Concepts

Virtual Warehouse

Virtual Warehouses

The term virtual warehouse in Snowflake catches out a lot of people who are new to Snowflake. This is because many of us instantly associate the word 'warehouse' with a 'data warehouse'—a physical repository of core business data to run dashboards or analytics to help in decision-making.

You can think of a virtual warehouse like a bundle of compute resources (CPU and RAM) in various sizes. These sizes are referred to as t-shirt sizes because they come in x-small, small, medium, large, and so on.

A virtual warehouse provides the compute (CPU and RAM) to work with the data—this includes loading data, unloading data, and pretty much performing any data manipulation language (DML) SQL queries against the data.

Virtual warehouse sizes and consumption

Virtual warehouse sizes and credit consumption

Warehouse Size	Credits/Hour
X-Small	1
Small	2
Medium	4
Large	8
X-Large	16
2X-Large	32
3X-Large	64
4X-Large	128
5X-Large*	256
6X-Large*	512

Warehouse Size	**Cost p/year(USD)**
x-Small	$ 1,800
Small	$ 3,600
Medium	$ 7,200
Large	$ 14,400
X-Large	$ 28,800
2 X-Large	$ 57,600
3X-Large	$ 115,200
4X-Large	$ 230,400
5X-Large	$ 460,800
6X-Large	$ 921,600

First minute is ***always*** billed for, e.g. 10 or 59 billed the same

When warehouse is suspended, no credits consumed

Scale Up vs Scale Out

Scale Up vs Scale Out

We've discussed the fact that Snowflake allows automatic scaling out by adding clusters to support high concurrency workloads, but it doesn't allow for automatic scaling up and down (between different ***virtual warehouse*** sizes) in the same way.

Therefore, you always must try to match your workloads to the size of your warehouse by performance testing.

This can lead to challenges when executing unpredictable workloads, which include short-running queries that complete within seconds alongside others that need to scan billions of rows and take minutes or even hours to complete.

A *p*

Query Acceleration Service

Query Acceleration Service

QAS is baked into each ***virtual warehouse*** to help alleviate this scenario, which allows Snowflake to offload large table scans to the QAS.

Currently, only the table scan, filter, and join-filter operations will benefit from query acceleration, but on warehouses used for executing unpredictable workloads, the performance gains can be significant.

Ultimately, the ***QAS*** attempts to find the 'right fit' of resources for those workloads with a mixture of unpredictable short-running queries and large, intensive queries.

On a small warehouse size this could result in a lot of queuing of smaller queries, while on an X-large warehouse those same queries wouldn't take advantage of the 128 CPUs available.

Provisions additional resources

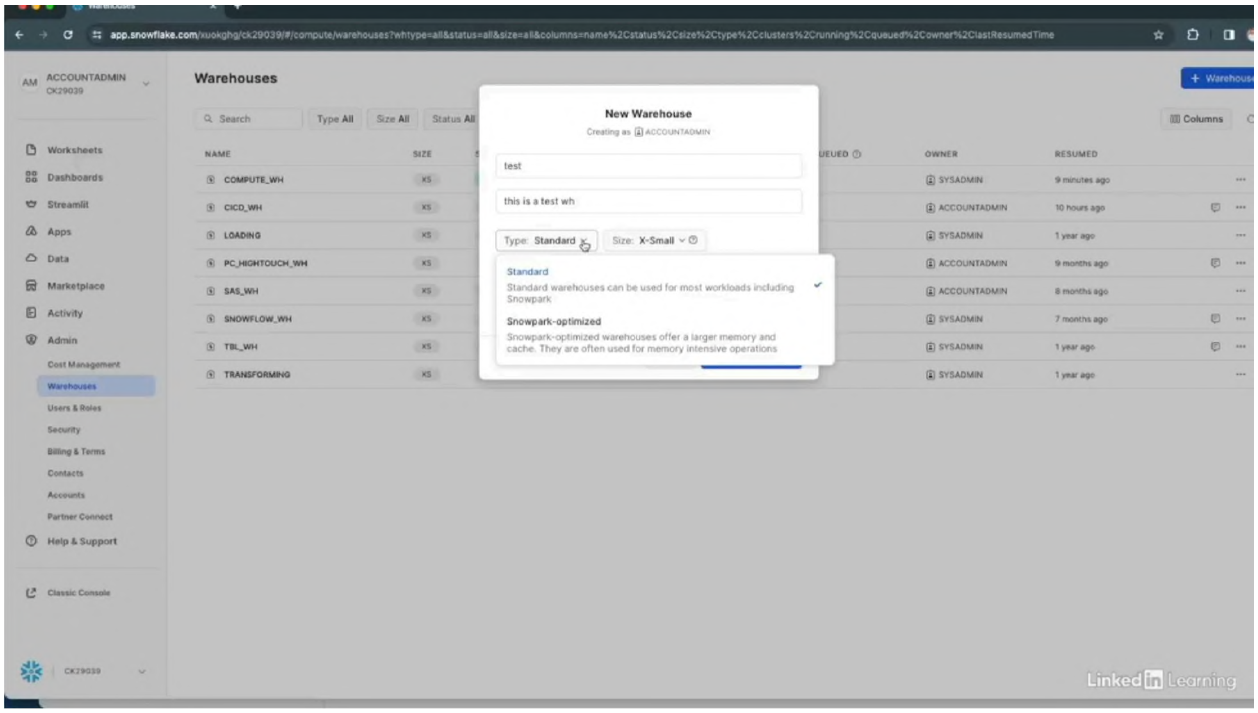
Query Acceleration Service

```
ALTER WAREHOUSE my_wh
E N AB LE_QU E RY_ACC LE RATIO N_S E RVIC
E =TRUE
QUERY_ACCLERATION_MAX_SCALE_FAC
TOR = 4;
```

This query allows Snowflake to automatically provision up to four times the size of the ***virtual warehouse*** it's attached to. This diagram illustrates how QAS can work to support an X-small warehouse that is receiving a variety of queries from Power BI. In this scenario, the QAS has been set to four times the size of the warehouse.

AAA

Configuring a virtual warehouse


app.snowflake.com/xuokghg/ck29039/#/compute/warehouses?whtype=all&status=all&size=all&columns=name%2Cstatus%2Csize%2Ctype%2Cclusters%2Crunning%2Cqueued%2Cowner%2ClastResumedTime

Warehouses

NAME	SIZE	OWNER	RESUMED
COMPUTE_WH	XS	SYSADMIN	9 minutes ago
CICD_WH	XS	ACCOUNTADMIN	10 hours ago
LOADING	XS	SYSADMIN	1 year ago
PC_HIGHTOUCH_WH	XS	ACCOUNTADMIN	9 months ago
SAS_WH	XS	ACCOUNTADMIN	8 months ago
SNOWFLOW_WH	XS	SYSADMIN	7 months ago
TBL_WH	XS	SYSADMIN	1 year ago
TRANSFORMING	XS	SYSADMIN	1 year ago

New Warehouse

Creating as ACCOUNTADMIN

test

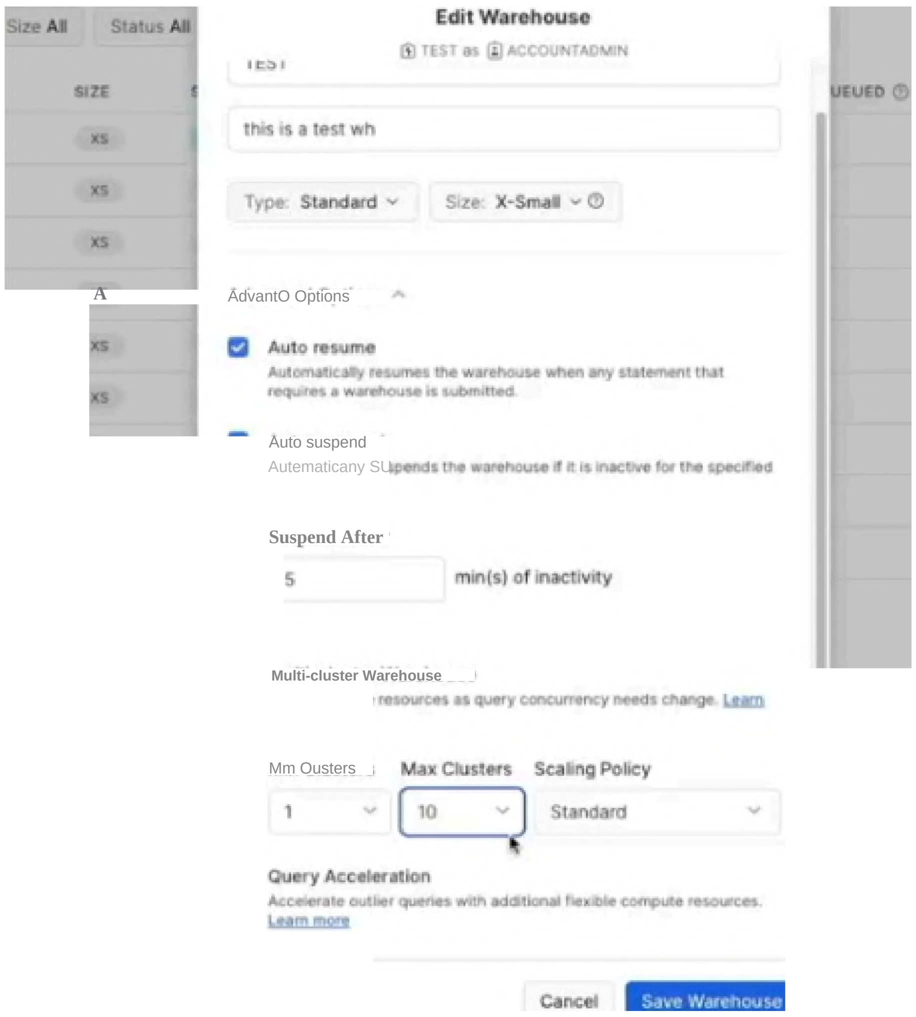
this is a test wh

Type: Standard

Size: X-Small

Standard ✓
Standard warehouses can be used for most workloads including Snowpark

Snowpark-optimized
Snowpark-optimized warehouses offer a larger memory and cache. They are often used for memory intensive operations



Edit Warehouse

TEST as ACCOUNTADMIN

TEST

this is a test wh

Type: Standard

Size: X-Small

AdvantO Options

Auto resume

Automatically resumes the warehouse when any statement that requires a warehouse is submitted.

Auto suspend

Autematicany SUspends the warehouse if it is inactive for the specified

Suspend After

5 min(s) of inactivity

Multi-cluster Warehouse

resources as query concurrency needs change. Learn

Mm Ousters	Max Clusters	Scaling Policy
1	10	Standard

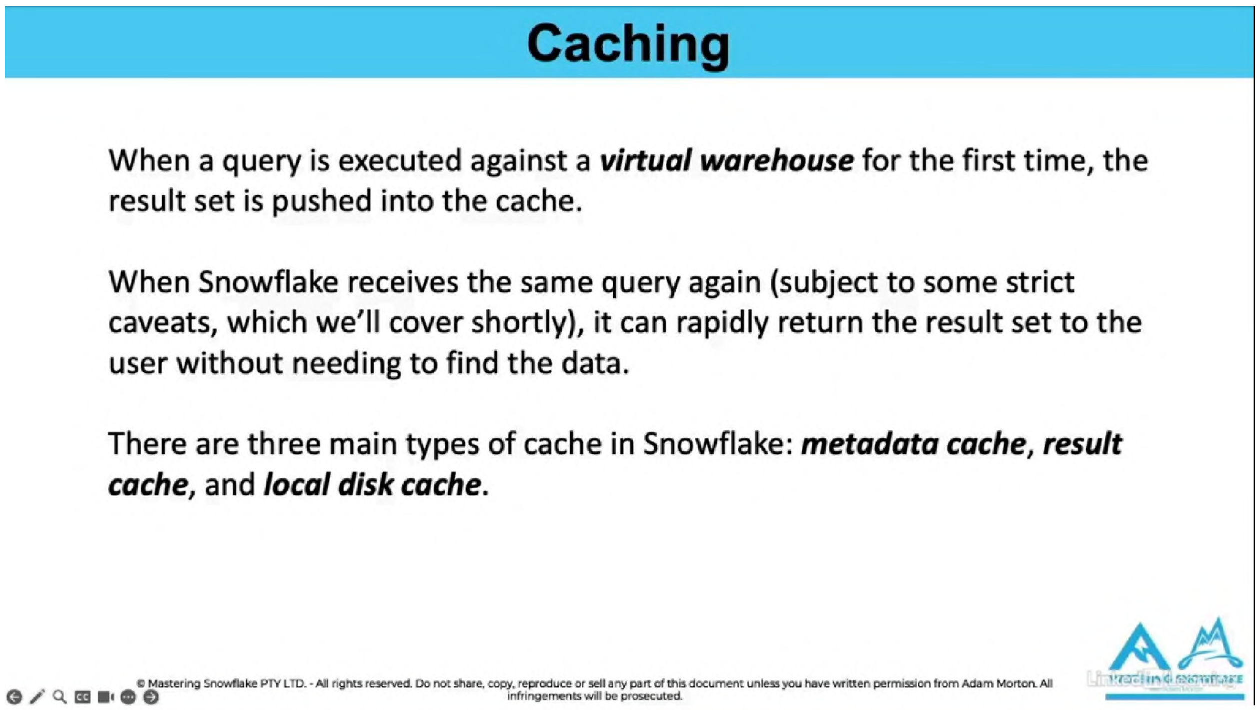
Query Acceleration

Accelerate outlier queries with additional flexible compute resources. Learn more

Default 5 minutes of inactivity for suspension of warehouse
Scale factor

Caching

- Reducing consumption cost because may not need WH to function



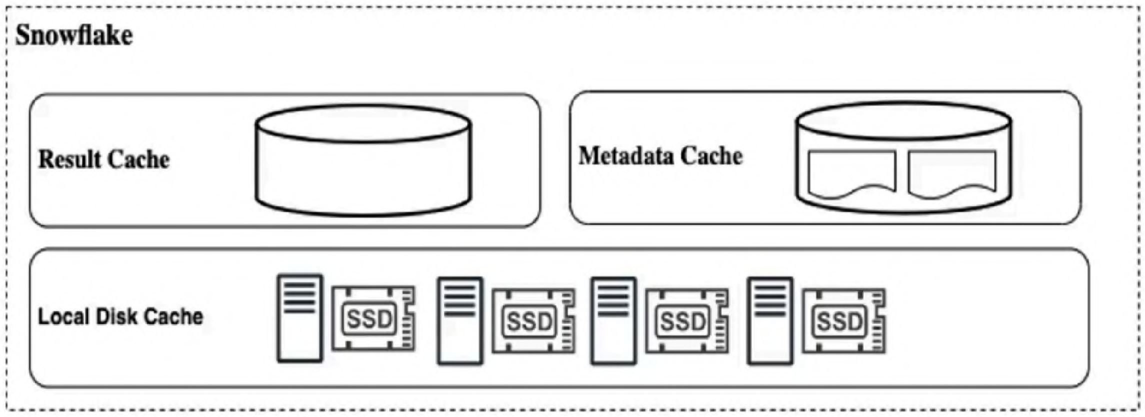
Caching

When a query is executed against a ***virtual warehouse*** for the first time, the result set is pushed into the cache.

When Snowflake receives the same query again (subject to some strict caveats, which we'll cover shortly), it can rapidly return the result set to the user without needing to find the data.

There are three main types of cache in Snowflake: ***metadata cache***, ***result cache***, and ***local disk cache***.

Caching



Metadata Cache

The ***metadata cache*** stores metadata about the objects and usage across your Snowflake account. This includes information about ***database objects,*** such as ***tables, views, stages,*** and ***micro-partitions,*** as well as information on which users are accessing tables and the history of queries within your account.

```
SELECT COUNT®*) FROM MYTABLE;
```

When you execute a query like the one shown above, you'll notice that if your ***virtual warehouse*** is suspended (or not set within the query). Snowflake won't ***auto resume*** the warehouse, nor will it prompt you for the warehouse name. This is because it doesn't require a ***virtual warehouse*** to provide the result set to the query; it is readily accessible in the ***metadata cache.***

AI

Result Cache

The ***result cache*** stores the result set of every query executed in the past 24 hours. If a user submits a subsequent query that matches the previous query, an attempt to retrieve the result set from the ***result cache*** is made.

There are some caveats to this to consider:

If the underlying data—which makes up the result set—changes, this will invalidate the result set held in the cache. Snowflake is aware of this because it knows what ***micro-partitions*** make up the result set. If any of the micro-partitions change, the result set held in the cache is dropped.

AA

Result Cache

If the query contains a function that needs to be evaluated at execution time (referred to as non-deterministic), the cache cannot be used. Examples here are CURRENT_TIMESTAMP or CURRENT_DATE.

The users executing the query must all have the correct ***privileges*** for all the tables used in the query.

Note: The ***result cache*** is not specific to an individual ***virtual warehouse—***it serves the whole environment. Any query by any user on the account that fits the criteria mentioned in the section above can take advantage of the result cache, regardless of the virtual warehouse they are using.

Suspending virtual warehouse will clear out result cache

Local Disk Cache

The ***local disk cache*** refers to the ***cache*** linked to a specific virtual warehouse SSD (solid-state drive).

When you execute a query for the first time, Snowflake will locate the data on disk and store all, or some (depending on the size of the result set), of that data in the ***local disk cache*** of the ***virtual warehouse*** that is providing the compute resources for the query.

Next time you execute the query, which can be either fully or partially satisfied by data in the local disk cache, Snowflake will leverage this cache, saving execution time.

Management and Monitoring

There are two ways of viewing ***virtual warehouse*** usage in the Snowflake web UI, depending on the version of the web UI you are using.

In Snowflake you'll have to switch to the Account Admin role. You can do this by clicking your username and selecting 'ACCOUNTADMIN' from the available roles

In ***Snowsight,*** you then head to Admin > Usage

Resource Monitors

Resource monitors in Snowflake allow you to configure actions in response to certain usage thresholds being hit at either the account or the warehouse level.

Your storage is often very inexpensive in comparison to compute; therefore, your focus should be centered on the compute resources so you can control the spend more closely.

To create a ***resource monitor,*** you will need to use the ACCOUNTADMIN role. This is required initially to set it up, but once created you can grant access to lower-level roles.

Resource Monitors

Resource monitors are typically set at the monthly level—in-line with Snowflake's billing cycle—although you can go all the way down to daily if you wish.

The credit quota is the number of credits to be used as the limit. Once this threshold is met, you can select certain actions in response. At the start of a new month (or whichever frequency you have selected), this will re-start at zero.

Resource Monitors

When your threshold is met—referred to as a trigger—you have three choices for how to respond:

Notify: You can be notified when a certain trigger is met. No other action is taken. Important: you must have enabled notifications within the WebUI for your account to receive these notifications!

Suspend: This will not accept any new queries, but it will allow any currently executing queries to complete. This behavior may well result in your exceeding your threshold, which very much depends on the active workload on your warehouse when the trigger is met.

Immediate: This is the only option that guarantees you do not go beyond your threshold. It's the most defensive option and will kill any queries that are currently executing on your warehouse.

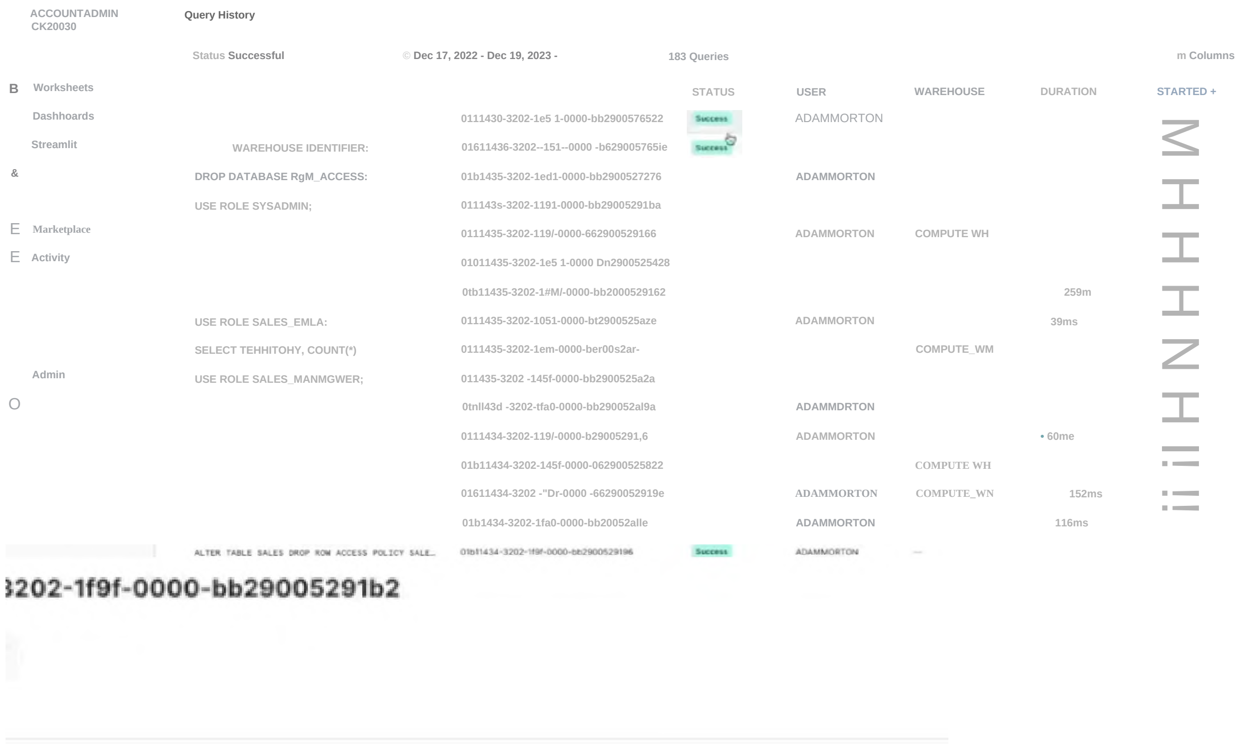
Query Optimization

The query profile provides a graphical representation of the query execution plan. The execution plan is a series of steps Snowflake undertakes to retrieve and process the data to satisfy a query.

Within the ***information schema*** there are a number of ***table functions*** we can use to identify those queries we might want to look at more closely. These functions are detailed in table on the next slide.

Query History Table Functions

Function Name	Description
QUERY_HISTORY	Returns queries within a specified time range.
QUERY_HISTORY_BY_SESSION	Returns queries within a specified session and time range.
QUERY_HISTORY_BY_USER	Returns queries submitted by a specified user within a specified time range.
QUERY_HISTORY_BY_WAREHOUSE	Returns queries executed by a specified warehouse within a specified time range.



ACCOUNTADMIN
CK20030

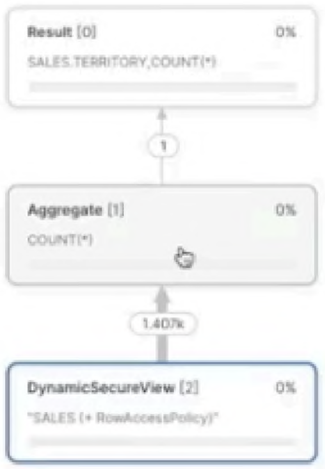
Query History

Status Successful   Dec 17, 2022 - Dec 19, 2023 -   183 Queries   m Columns

B Worksheets
Dashhoards
Streamlit
&
E Marketplace
E Activity
Admin
O

		STATUS	USER	WAREHOUSE	DURATION	STARTED +
	0111430-3202-1e5 1-0000-bb2900576522	Success	ADAMMORTON			
WAREHOUSE IDENTIFIER:	01611436-3202--151--0000 -b629005765ie	Success				
DROP DATABASE RgM_ACCESS:	01b1435-3202-1ed1-0000-bb2900527276		ADAMMORTON			
USE ROLE SYSADMIN;	011143s-3202-1191-0000-bb29005291ba					
	0111435-3202-119/-0000-662900529166		ADAMMORTON	COMPUTE WH		
	01011435-3202-1e5 1-0000 Dn2900525428					
	0tb11435-3202-1#M/-0000-bb2000529162				259m	
USE ROLE SALES_EMLA:	0111435-3202-1051-0000-bt2900525aze		ADAMMORTON		39ms	
SELECT TEHHITOHY, COUNT(*)	0111435-3202-1em-0000-ber00s2ar-			COMPUTE_WM		
USE ROLE SALES_MANMGWER;	011435-3202 -145f-0000-bb2900525a2a					
	0tnll43d -3202-tfa0-0000-bb290052al9a		ADAMMDRTON			
	0111434-3202-119/-0000-b29005291,6		ADAMMORTON		• 60me	
	01b11434-3202-145f-0000-062900525822			COMPUTE WH		
	01611434-3202 -"Dr-0000 -66290052919e		ADAMMORTON	COMPUTE_WN	152ms	
	01b1434-3202-1fa0-0000-bb20052alle		ADAMMORTON		116ms	
ALTER TABLE SALES DROP ROW ACCESS POLICY SALE…	01b11434-3202-1f9f-0000-bb2900529196	Success	ADAMMORTON	—		

3202-1f9f-0000-bb29005291b2



01b11431-3202-1ed1-0000-bb29005271ba

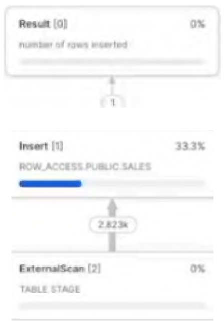
A ADAMMORTON

Most Expensive Nodes 1 of 3)

Insert ni nn

Profile Overview (shed)

Total Execution Time (595ms) 100 o*	
- Remote Disk vO	333%
• initialization	66 7%


Result [0] 0%
number of rows inserted
1
Insert [1] 33.3%
ROW_ACCESS.PUBLIC.SALES
2.823k
ExternalScan [2] 0%
TABLE STAGE

Statistics

Number of rowsinterted	282
Sean progress	100 00%
External bytes scanned	0 41MB
Bytes written	GOSMB

Data Spilling

Data Spilling

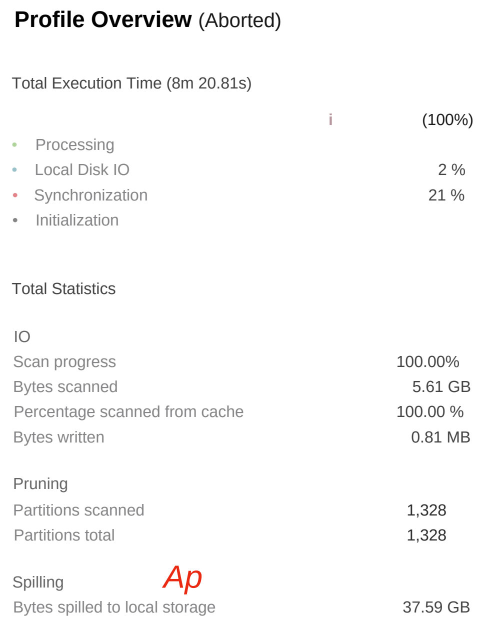
Data spilling indicates the memory in your ***virtual warehouse*** isn't large enough to service your workload.

When this happens, data needs to be processed somewhere, and if there's no memory left within your warehouse, the disk is the next option.

The first level is the local disk storage, which is the SSD cache, followed by the remote disk, which is the most inefficient place to process data.

When data spills, files relating to your dataset are created and retrieved on disk behind the scenes, which is grossly inefficient in comparison to working with data in memory.

Data Spilling


Profile Overview (Aborted)

Total Execution Time (8m 20.81s)

i (100%)

• Processing	
• Local Disk IO	2 %
• Synchronization	21 %
• Initialization	

Total Statistics

IO	
Scan progress	100.00%
Bytes scanned	5.61 GB
Percentage scanned from cache	100.00 %
Bytes written	0.81 MB
Pruning	
Partitions scanned	1,328
Partitions total	1,328
Spilling	
Bytes spilled to local storage	37.59 GB

Ap

'Bytes spilled to local storage' the data can fit within the ***local disk cache.***

'Bytes spilled to remote storage' is the next level where there's not enough local storage.

Ideally, you'd want to avoid both scenarios, but primarily you should look out for anything that is spilling to remote storage, as that is significantly slower than data spilling to local storage.

Micro-partition pruning

Micro-Partition Pruning

The ***micro-partitions*** are organized based on a ***clustering key,*** which is automatically selected by Snowflake and typically is based on the order the data is loaded into Snowflake.

It is a fact that Snowflake's architecture is underpinned by these small blocks of data, which gives it a distinct performance advantage in many cases.

As the data is organized across these small blocks of ***micro-partitions,*** we could potentially have thousands of these partitions for a large table. This allows for more flexibility when locating and retrieving the data to satisfy a query.

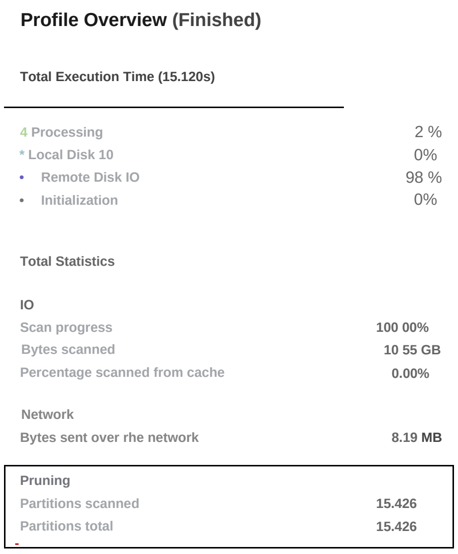
Micro-Partition Pruning

When a user executes a query with a filter, such as a WHERE clause, Snowflake will use the metadata associated with the ***micro-partitions*** to read only those that should contain the required data.

Taking this approach means that time isn't spent unnecessarily reading through micro-partitions that don't contain the data.

This concept is known as ***query pruning,*** and it's very important to understand this concept before heading into the certification exam.

Micro-Partition Pruning


Profile Overview (Finished)

Total Execution Time (15.120s)

4 Processing	2 %
* Local Disk 10	0%
• Remote Disk IO	98 %
• Initialization	0%

Total Statistics

IO

Scan progress	100 00%
Bytes scanned	10 55 GB
Percentage scanned from cache	0.00%

Network

Bytes sent over rhe network	8.19 MB

Pruning

Partitions scanned	15.426
Partitions total	15.426

Using the ***query profile*** we can see the number of partitions scanned against a total number of partitions.

Ideally, we want to see the query profile telling us that we're scanning a small percentage of the total available partitions.

AA

Domain 3 - Recap

- You can identify the efficiency of query pruning using Snowflake's query plan and looking at total partitions and partitions scanned.
- The 'Bytes spilled to local storage' metric in the query profile can be used to understand if the result set is too large to fit into warehouse memory.
- The maximum amount of server clusters in a warehouse is 10.
- When a warehouse is resized only new queries will take advantage of the additional resources.
- Users must wait until Snowflake provisions all servers in a virtual warehouse before it becomes available to execute queries.

Search Optimization Service

Search Optimization Service

if the filter is on something other than the ***clustering key,*** the ***query optimizer*** may well have to scan all the available partitions, resulting in a query that performs poorly.

To help with these specific-use cases, Snowflake introduced a feature called ***search optimization service.*** This can be activated at a table level, as shown in the query below:

```
alter table test_table add search optimization;
```

costly

External Stages

When you create an external stage in Snowflake, it acts as a pointer to a third-party cloud storage location.

This storage location can be Amazon S3, Google Cloud Storage, or Microsoft Azure, regardless of what platform you run your Snowflake account on.

So, for example, you can run your Snowflake account on Amazon and create an external stage on Google Cloud Storage or Microsoft Azure.

Creating an External Stage

Creating an external stage in Snowflake creates a database object within the selected schema.

This object holds the URL (to the cloud storage location) along with any security credentials to access the required cloud storage location.

When you drop an external stage, all you are doing is removing the pointer. Therefore, the files in the external stage remain unaffected.

External Tables within an External Stage

Contained within your external stage, you can have external tables. These objects hold metadata, which tells Snowflake where to locate those data files that relate to the table.

The big tradeoff here is performance. Because data is not in Snowflake's native, compressed, micro-partitioned format, the same approach to efficient query pruning across micro-partitions while leveraging metadata cannot be realized.

Refreshing External Tables within an External Stage

There is a small maintenance overhead charge for manually refreshing the external table metadata using the ALTER EXTERNAL TABLE REFRESH command.

It is possible to refresh this table metadata automatically using the corresponding cloud notification service, such as Amazon's Simple Queue Service (SQS) or Azure Event Grid.

Internal Stages

USER Stage

The user stage is allocated to each Snowflake user. This is when only one user needs to access the staging data before loading it into multiple target tables. You refer to a user stage with the following prefix @~.

```
copy into mytable from @~/staged file_format = (format_name =
'my_csv_format');
```

AAA

Internal Stages

TABLE Stage

Each table has a table stage associated with it. The name of the table stage is always the same as the table it relates to. Multiple users can access the data files, but these files can only load into one target table. You refer to a table stage with the following prefix @%. You can list the files in a table stage as follows:

```
LIST @%my_table
```

A .A

Internal Stages

NAMED

A named internal stage refers to an area where all the data files exist within Snowflake. Multiple users can access data in this stage, while the data files can also target multiple tables. You refer to a named internal stage with the following prefix @.

```
create or replace stage my_stage
  file_format = my_csv_format;
copy into mytable from @my_stage;
```

A M

File Formats

File Formats

When it comes to loading or unloading structured or semi-structured data, a file format tells Snowflake how to read the data.

For example, when working with a CSV (comma-separated values) file, the file format object allows you to specify options, such as what kind of delimiter the file uses, how to treat empty fields, whether you need to skip a header row, and what happens if the column count doesn't match the number of columns in the target table.

Supported File Formats

Format	Type	Load	Unload
CSV	Structured	Yes	Yes
AVRO	Semi-structured	Yes	No
JSON	Semi-structured	Yes	Yes
ORC	Semi-structured	Yes	No
Parquet	Semi-structured	Yes	Yes
XML	Semi-structured	Yes	No

File Formats

You can define your file format on the fly by specifying it alongside your command to load or unload the data, or you can create it ahead of time and give it a name.

A named file format exists as an object in the database, meaning not only can you reuse it, but other users can also reference the same file format, providing they have the correct permissions.

File Formats

The code snippet below shows the basic syntax to create a named file format:

```
CREATE [ OR REPLACE ] FILE FORMAT [ IF NOT EXISTS ] <name>

                    TYPE = { CSV ] JSON | AVRO | ORC | PARQUET | XML
} [ formatTypeOptions ]

                        [ COMMENT = '<string_literal>' ]
```

PUT Command

Firsts to stage your data from a local folder on a machine to an internal stage, use the PUT command as follows:

```
put file://c:\temp\data\mydata.csv @~ auto_compress=true;
```

COPY INTO Command

Once you have staged your data in either an external or internal stage, you can load your data into a pre-existing target table. Use the COPY INTO command.

The COPY INTO command also supports some basic transformations of the data, including:

- Re-ordering of columns
- Omission of columns
- Casting of data types

COPY INTO Command

However, you cannot filter the data as part of the COPY INTO command by using a WHERE clause or LIMIT.

When using a COPY INTO command, you can reference the named file format when you load or unload your data as follows:

```
copy into mytable
  from s3://mybucket/data/files
  storage_integration = myint
  file_format = (format_name = my_csv_format);
```

Insert Command

Once your data is in a table in Snowflake, you can use the INSERT command to insert one or more rows into a target table. This is like any other database system you may have used in the past.

Alternatively, you can choose to truncate the table as part of your INSERT statement using the OVERWRITE command.

```
insert overwrite into sf_employees
  select * from employees
```

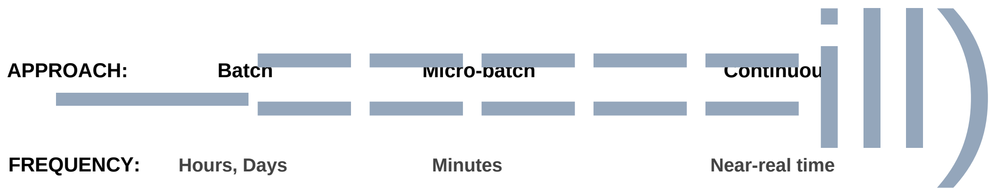
GET Command

If you wish to download your data from an internal stage to your local machine, you can use the GET command.

```
get @~/myfiles file:///tmp/data/;
```

Bulk vs. continuous loading

Bulk versus Continuous Loading



In the batch approach, data accumulates over time (hours, days) and is then loaded periodically. While the continuous data loading approach aims to ensure every data item is loaded as it arrives, the happy medium between the two is an approach called micro-batch, which allows data to accumulate over small time windows (minutes) before being loaded.

Introduction to Snowpipe

Snowpipe is a fully managed service. It doesn't require a user-defined virtual warehouse because it uses managed compute resources provided for you transparently.

Snowpipe is designed to move smaller amounts of structured or semi-structured data quickly and efficiently from a stage to a table in Snowflake.

You must create and stage the files first, which means you cannot stream data directly from the source system into Snowflake.

Introduction to Snowpipe

Snowpipe is a COPY INTO command wrapped into a CREATE PIPE statement, as the following code sample illustrates:

```
create pipe mypipe as copy into mytable from @mystage;

create pipe mypipe2 as copy into mytable(cl, c2) from (select $5, $4 from @mystage);
```

Introduction to Snowpipe

Snowpipe removes the need for tuning or any other additional management, and you don't need to worry about the resources because it's completely serverless.

This means you don't need to manage the underlying infrastructure or be concerned with scaling up or down. You only pay for the compute time used to load data.

You still pay using Snowflake credits. Even though there's no user-defined virtual warehouse required, it is still listed on the Snowflake bill under a separate virtual warehouse named Snowpipe.

A A

Introduction to Snowpipe

Snowpipe is a COPY INTO command wrapped into a CREATE PIPE statement, as the following code sample illustrates:

```
create pipe mypipe as copy into mytable from @mystage;

create pipe mypipe2 as copy into mytable(cl, c2) from (select $5, $4 from @mystage);

create pipe mypipe_s3
  autojngest = true
  aws_sns_topic = ‘arn:aws:sns:us west-2:001234567890:s3_mybucket‘
 as
  copy into snowpipe_db.public.mytable
  from @snowpipe_db.public.mystage
  file_format = (type = 'JSON');
```

- Pipe is realtime
- Smaller files
- Completely serverless in the background
- ***COPY INTO statement stored for 64 days***
- ***Load history for metadata stored for 14 days***

Snowpipe Billing

With Snowpipe's serverless compute model, users can initiate any size load without managing a virtual warehouse.

Instead, Snowflake provides and manages the compute resources, automatically scaling up or down based on the current Snowpipe load.

Accounts are charged based on their actual compute-resource usage, in contrast with customer-managed virtual warehouses, which consume credits when active and may sit idle or be overutilized.

Snowpipe Billing

Account administrators or users with the MONITOR USAGE global privilege can use Snowsight, the Classic Web UI, or SQL to view the credits billed to your Snowflake account within a specified date range.

Method	Usage
Snowsight	**Select Admin > Usage**
Classic Web UI	**Select Account > Billing & Usage**
SQL	**Query either:** • **PIPE_USAGE_HISTORY table function (in the Snowflake Information Schema)** • **PIPE_USAGE_HISTORY view (in Account Usage)**

Best practices for data loading

File Sizes

Snowflake recommends that you aim to load data files of roughly 100-250 MB or larger (compressed) and stage files at a minimum of one-minute intervals for maximum efficiency.

Breaking larger files into multiple, smaller files improves performance by allowing Snowflake to distribute the load among the compute resources in the active virtual warehouse. You may want to consider aggregating smaller files to minimize the processing overhead for each file.

Folders

Both internal and external stage references can include a path to where your staged files live.

When staging datasets as part of a regular data-load process. Snowflake recommends that you partition the data into logical paths and note identifying details such as geographical location or other source identifiers, along with the date when the data was written.

File Partitioning

For example, if you were storing data for an international company by geographical location, you might include identifiers such as country and city in paths, along with data write dates, such as:

- Canada/Toronto/2022/07/03
- Japan/Tokyo/2022/07/03
- United_States/New_York/2022/07/03

Validation Mode

The VALIDATION MODE command allows you to check the validity of a COPY INTO command without loading the data into Snowflake. This comes in handy during testing.

Let's say you are provided with 100 files to load into Snowflake for the first time

You can save time and get some early feedback by parsing the data in the files, just by adding the VALIDATION MODE to the COPY INTO statement.

```
copy into mytable validation_mode = ‘RETURN_ERRORS‘;
```

Error on column count mismatch

If the number of delimited columns in an input data file does not match the number of columns in the corresponding table, Snowflake will, by default, throw an error.

Sometimes, you might need to override this behavior if, for example, you know your file has fewer columns than your target table.

To do this, you can use the ERROR_ON_COLUMN_COUNT_MISMATCH copy option and set it to equal FALSE. This will mean the file will be successfully loaded if the input file has more (or fewer) columns than the target table. Snowflake will load the columns in order and disregard any that remain.

Error-handling options - ON_ERROR

Supported Value	Description
CONTINUE	This option will continue loading the file, regardless of errors found.
SKIPFILE	If an error is found in the file, the entire file will be skipped. If you are loading one file, then the load process for that file will be aborted. If you are loading multiple files, Snowflake will move to the next file in the process.
ABORT_STATEMENT	The entire load process will be aborted if any errors are found.

The default value for bulk loading using COPY INTO is ABORT_STATEMENT, while the default for Snowpipe is SKIP_FILE

Unloading Data

Unloading Data

As we touched on earlier, it is possible to unload data from a relational table in Snowflake to a CSV, JSON, or Parquet file format.

To unload data from a table in Snowflake, you still use the COPY INTO command, even though data is going out of Snowflake, not into it. The results of the query are written to one or more files as specified in the command, and the file(s) are stored in the specified location, which could be an internal or external stage.

Unloading Data

The following statement joins and unloads data from two tables, dim_account_items and fct_lmi_account_items, into the named internal stage my_stage while limiting the result set to 100 records:

```
copy into @my_stage/result/data_
from
(select *
from DIM ACCOUNT ITEMS dim
inner join FCT_LMI_ACCOUNTJTEMS fct
on dim.accjd = fct.acc_id limit 100);
```

Unloading Data

After the statement has run successfully, we can use the following command to view the file in the stage:

```
List @my_stage;
```

```
name                                 size   md5                                last_modified
my_stage/result/data_0_0_0.csv.gz   2,064   7cf89c098ec1cb6f4fe7fc22e2db45a3   Mon, 18 Jul 2022 08:16:18 GMT
```

Unloading into multiple or single files

The COPY INTO command provides an option called 'Single' for unloading data into a single file or multiple files.

The default is SINGLE = FALSE, which will unload into multiple files.

When unloading data into multiple files, you can use the MAX_FILE_SIZE copy option to specify the maximum size of each file created, as the following code snippet shows:

```
copy into @my_stage from mytable max_file_size = 49000000;
```

Note: Maximum file size is 5GB for cloud storage

Unloading into multiple or single files

To unload data to a single output file at the potential cost of decreased performance, specify the SINGLE = true copy option in your statement:

```
copy into @my_stage from mytable single=true;
```

Handling empty strings and null values

To control this behavior, you can specify certain file format options. There's an option called 'field_optionally_enclosed_by,' where you can use a single or double quote to tell Snowflake how strings will be enclosed.

This is still optional as Snowflake will allow you to unload empty string values without enclosing quotes.

There is also another file format option called 'empty_field_as_null.' This accepts a Boolean value (TRUE or FALSE).

If you were to leave the field_optionally_enclosed_by option out, the default would be none, meaning the string fields would be enclosed. Then, by setting empty _field_as_null to FALSE, all empty strings would be unloaded as empty fields. This approach is not recommended, as you wouldn't be able to distinguish between emptystrings and NULL values.

Streams and tasks

Introduction to Streams

In Snowflake, when you create a stream, two main things happen. First, a pair of hidden columns are added to the stream, which begins to store change-tracking metadata.

Second, and at the same time, a snapshot of the source table is logically created. This snapshot acts as a baseline so that all subsequent changes in the data can be easily identified.

Stream Metadata Columns

Metadata Column	Description
METADATASACTION	This tells you what DML action was performed (INSERT or DELETE). Note: An update is effectively a DELETE followed by an INSERT.
METADATA$ISUPDATE	A Boolean value, which indicates if the records were part of an UPDATE operation. When TRUE, you should expect to see a pair of records, one with a DELETE, and one with an INSERT. Note: The stream provides you with a net change between two offset positions. So, if a record were inserted and subsequently updated within the offsets, this will be represented just as a new row with the latest values. In this case, the field will be FALSE.
METADATASROW_ID	A unique ID for the row. This can be very helpful for row-level logging and auditability throughout your system. I would recommend capturing this and storing it as part of your solution so you can accurately track the flow of data through your system.

Introduction to Tasks

A task allows you to execute a single SQL statement on a schedule or on demand. If you wish to execute multiple 5Q.L statements, you'll need to define a task for each. It's then possible to chain these tasks together in a sequence.

You must, however, start with a master or parent task, which has the job of being executed first in the sequence.

A basic task could look something like this:

```
CREATE TASK mytask_minute
  WAREHOUSE = mywh
  SCHEDULE = ’5 MINUTE’
AS
INSERT INTO mytable(ts) VALUES(CURRENT_TIMESTAMP);
```

Introduction to Tasks

It is then possible to chain tasks together in a sequence, as the following example illustrates:

```
create task tasks
  after task2
as
insert into tl(ts) values(current_timestamp);
```

SYSTEM$STREAM_HAS_DATA

A typical pattern would be to have a task running that checks the stream for the presence of records to be processed using the system function SYSTEM$STREAM_HAS_DATA('<stream_name>').

If the function returns FALSE, then there are no records to process, and the task will exit. If the function returns TRUE, there are new records to be consumed.

Domain 4.0 - Recap

- In a Snowpipe load process, data follows the Pipe > Stage > Table sequence as it arrives in Snowflake
- When deciding whether to use bulk loading or Snowpipe, you should consider how often you need to load the data and the number of files to be loaded at any one time.
- When loading data into Snowflake from an external stage you can select a subset of the data.
- A stage in Snowflake is where files rest before they are loaded into a table in Snowflake

Domain 4.0 - Recap

- The load metadata stored for each table into which data is loaded into Snowflake is kept for 64 days.
- It is possible to load data into a table if the number of columns don't match.
- If you mistakenly attempt to load the same file into a table in Snowflake that had already been successfully loaded the previous day nothing will happen as Snowflake will ignore the file.

Domain 4.0 - Recap

- The load metadata stored for each table into which data is loaded into Snowflake is kept for 64 days.
- It is possible to load data into a table if the number of columns don't match.
- If you mistakenly attempt to load the same file into a table in Snowflake that had already been successfully loaded the previous day nothing will happen as Snowflake will ignore the file.
- Snowpipe is a feature offered by Snowflake to load data continuously.

Domain 4.0 - Recap

- When using the COPY INTO command you can aggregate columns, cast columns, concatenate columns and re-order columns but not filter data.

- You are loading a file into Snowflake using the COPY INTO command. One of the rows in the source data is causing an error when loading. You would best address the issue of loading all the other records by using the COPY option ON_ERROR = CONTINUE.

Domain 4.0 - Recap

- You are working on implementing a new Snowflake solution for a client. You have been provided with a set of source files from your legacy on-premises data warehouse. You need to validate that these files are in the expected format before you attempt to load them into your Snowflake environment. You'd do this by running the the COPY INTO command using the VALIDATION_MODE option.

Domain 4.0 - Recap

- Snowpipe's serverless compute model is based on bytes transferred.

- Recently you have noticed your billing costs have increased. You think this is related to implementing Snowpipe for continuous data loads. To view the costs relating to Snowpipe you can either head to Admin > Cost Management, or in SQL query the PIPE_USAGE_HISTORY table function (in the Snowflake Information Schema) or the PIPE_USAGE_HISTORY View (in Account Usage).

Domain 5.0: Data Transformations

Estimating Functions

Snowflake’s estimating functions

Snowflake's estimating functions can be broken down into four broad groups:

- Cardinality estimation
- Similarity estimation
- Frequency estimation
- Percentage estimation

CARDINALITY ESTIMATION

When it comes to cardinality estimation, Snowflake leverages a state-of-the-art algorithm for some operations.

This algorithm is called HyperLogLog and can be used in place of count distinct operations.

This is because the amount of compute resources required is proportional to the cardinality in the dataset.

For extremely large datasets, this would become impractical and costly. HyperLogLog can determine cardinality at a fraction of this cost with a high degree of accuracy. For the exam, just having an awareness of HyperLogLog is enough,

SIMILARITY ESTIMATION

Snowflake uses MinHash for estimating the approximate similarity between two or more datasets.

The MinHash scheme compares sets without computing the intersection or union of the sets, which enables efficient and effective estimation.

You can use these functions to determine how similar different datasets might be.

FREQUENCY & PERCENTAGE ESTIMATION

FREQUENCY ESTIMATION

Snowflake uses the Space-Saving algorithm, a space-and-time-efficient way of estimating approximate frequent values in datasets.

PERCENTAGE ESTIMATION

Snowflake uses the t-Digest algorithm, a space and time-efficient way of estimating approximate percentile values in datasets.

Table Sampling

The two main methods for sampling data in Snowflake are:

1. You can specify a fraction of a table along with a specified probability for an individual row. You can provide a number from 0 to 100 to specify the percentage probability from the available data in the table to use for selecting the sample. The number of rows returned depends on the size of the table and requested probability.

2. You can sample a fixed number of rows from a table. The rows requested are returned unless the table contains fewer rows than requested.

Table Sampling

Use the keywords TABLESAMPLE or SAMPLE. To return a sample of a table in which each row has a 10% probability of being included in the sample, you can run the following code snippet:

```
select * from testtable sample (10);
```

Alternatively, you can use the fixed-size sampling method. In the example below, we return 10 rows from the table:

```
select * from testtable sample (10 rows);
```

Stored Procedure - Benefits

You can write a stored procedure in any of the following programming languages:

- JavaScript
- Snowflake scripting (for SQL)
- Java
- Python
- Scala

Stored Procedure - Benefits

- Handling errors gracefully and logging consistently
- Building an SQL statement at runtime before executing it
- Branching and looping logic—which SQL doesn't support, but the other available programming languages do.

User-defined functions

UDFs

In Snowflake, you can define a UDF in one of the four following languages: Python, Java, JavaScript, or SQL. Some of the factors that may influence your decision on which programming language you use to write a UDF are:

• Whether you already have code in a particular language. For example, if you already have a .jar file containing a method that does the work you need, you might prefer Java as your programming language.

• The capabilities of the language. For example, SQL doesn't support looping and branching, while the other three languages allow for this approach.

• Whether a language has libraries that can help you do the processing that you need to do.

UDFs

All four languages support the two types of UDFs available in Snowflake:

- Scalar function—returns one output row for each input row.
- Table function—returns zero, one, or many records for each input row.

It is worth noting that, with a Java or Python UDF, you can provide a source file, such as a Jar file in a precompiled format for Java or as a staged file for Python. SQL and JavaScript, however, do not support this.

Working with semi-structured data

Working with Semi-Structured Data

To do this. Snowflake has introduced a special type of data known as a variant, which allows you to insert JSON, AVRO, Parquet, ORC, or XML data directly into this field without needing to tell Snowflake what the format of this data is upfront.

Make sure you keep in mind the five types of semi-structured data Snowflake supports (JSON, AVRO, Parquet, ORC, and XML) for the certification exam.

Note: The maximum length of a variant column is 16 MB

Working with Semi-Structured Data

```
create table variant_test (coll variant);
insert into variant_test
select ['London¹, 'Sydney' ];
select coll[l] from variant_test;
```

Here, we create a new table with one variant column. We then insert an array containing two values: London and Sydney. Next, we select a value from the array by specifying the position of the value. In this case, we want to select the value 'Sydney/ so we specify the position of 1.

Array data type

In addition to the variant data type. Snowflake also has object and array datatypes, which support working with semi-structured data.

A Snowflake array is like an array in many other programming languages An array contains 0 or more pieces of data.

Each element is accessed by specifying its position in the array.

Object data type

An object data type is comparable to a JSON object. Snowflake uses the object datatype to convert relational data to the JSON format easily.

One way of doing this is to use a function called object_construct(), which, as the name suggests, constructs an object from the data you input into the function.

To take a JSON-compliant variant and convert it to a string, you can use a function called TO_JSON(). Conversely, to take a string and convert it to a JSON-compliant variant, you should use a function called PARSE_JSON.

Querying Semi-Structured Data

You can query semi-structured data using special SQL extensions in your queries, which 'flatten' the data.

FLATTEN is a table function in Snowflake that takes a variant, object, or array and converts semi-structured data to a relational representation. In the following example, we provide a string '[1, 77]' to the parsejson function, which converts our string to a JSON object.

We then provide that JSON data type as an input into the flatten function to return a relational data type:

```
select * from table(flatten(input => parsejson('[l, ,77]'))) f;
```

Working with Unstructured Data

Working with Unstructured Data

You can store unstructured data files in either external or internal stages. A directory table catalogs a list of these staged files. You need to turn on directory tables when you create a stage using the following command:

```
alter stage my_stage_name
set directory = (enable = true);
```

You can also force a refresh of your directory table as follows:

```
alter stage my_stage_name refresh;
```

Once it has refreshed, you can then query the directory table to retrieve a URL that points to your unstructured data file, as the following example demonstrates:

```
SELECT * FROM DIRECTORY( @my_stage name );
```

Domain 5 - Recap

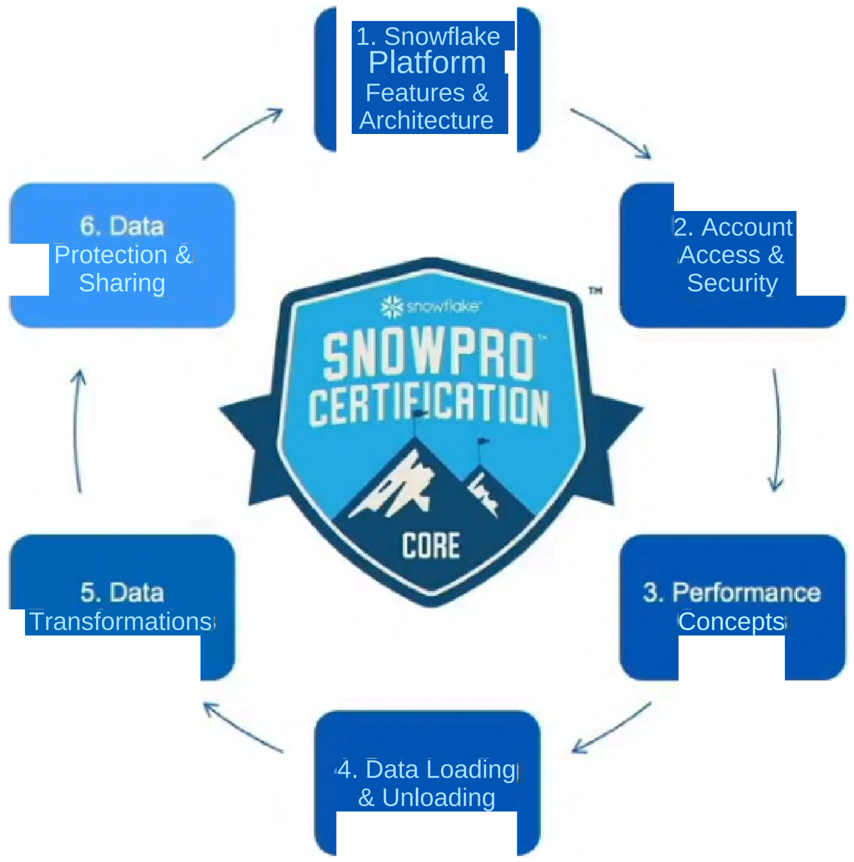
- You can load semi-structured data directly into a variant column in Snowflake.
- To make semi-structured data available to a BI tool, use a view with the FLATTEN function to present the data in a relational format.
- You can easily format the query results from a RELATIONAL table in Snowflake to a JSON format using the object_construct function.
- Use the STRIP_OUTER_ARRAY file format option for the COPY INTO <table> command to remove the outer array structure and load the records into separate table rows.

Domain 5 - Recap

- The maximum length of a VARIANT is 16 MB.
- Directory tables store a catalog of staged files in cloud storage. Roles with sufficient privileges can query a directory table to retrieve file URLs to access the staged files as well as other metadata.
- A directory table is not a separate database object; it is an implicit object layered on a stage. Both external (external cloud storage) and internal (i.e., Snowflake) stages support directory tables.

Domain 6.0: Data Protection

The SnowPro Core Domains


1. Snowflake Platform Features & Architecture
2. Account Access & Security
3. Performance Concepts
4. Data Loading & Unloading
5. Data Transformations
6. Data Protection & Sharing
snowflake
SNOWPRO CERTIFICATION
CORE

What you will learn

- Infrastructure Protection
- High Availability
- Data Encryption
- Data Protection
- Cloning
- Time Travel
- Fail-safe
- Storage Concepts
- Secure Data Shares
- Data Marketplace

Infrastructure Protection

Infrastructure Protection

Because the Snowflake architecture separates the compute, storage, and service layers, Snowflake assures resiliency and data consistency in the event of node failures. Snowflake also takes care of any node failures seamlessly, without the need for user intervention.

High Availability

High Availability

Snowflake provides standard failover protection across three availability zones. As yc ingest your data, it is replicated synchronously and transparently across availability zones. This protection is automatically extended from Snowflake to customers at no added charge. It also extends to the metadata captured by Snowflake.

With Snowflake, you can replicate data between multiple regions and across multiple cloud service providers too. This means you can have one Snowflake account in Goog replicating data to another account running on AWS. Therefore, if the Google service suffers a major outage, you can failover to your AWS standby service, allowing your business to continue to function as usual.

Note: Failover and tailback are features of Snowflake's Business Critical edition or higher.

- creates redundancy

Data Encryption

Data Encryption

Snowflake uses encryption for all customer communications to ensure data isn't sent over the network in clear text. Data at rest within Snowflake is always encrypted. This is the default configuration and cannot be switched off.

Note: Data within internal stages (within Snowflake) is encrypted by default.

If you opt to have an external stage—remember, this is a storage area on a cloud storage platform—you may choose to encrypt the data within this location. Encrypting the data in this way is known as 'client-side encryption.'

Snowflake will always encrypt data immediately when it stores it, regardless of whether this data arrives in an unencrypted or encrypted format.

Data Protection

Data Protection

While the underlying cloud provider provides standard cloud data protection, Snowflake takes this a stage further by applying a concept called time travel, which enables you to recover data from any point in time, up to 90 days.

In addition, it's all accomplished automatically. Other than specifying the number of days for data retention, you do not have to initiate a thing or manage snapshots.

Cloning (Zero-copy)

Cloning

The main reason for this is that there is no physical movement of data in the background.

It's a metadata operation, meaning when you clone an object it creates a copy of the object with a pointer back to the original data.

If a query changes the original data that the clone was based on after you created it, it will not affect the data in the clone. Similarly, you are free to manipulate the data in the clone and this won't affect the original object it was based on. It will, however, start to incur storage costs based on the data you have changed.

Cloning Benefits

The primary benefit of cloning is that it facilitates the creation of development or test environments as part of an overall development lifecycle approach.

Traditionally, you'd have had to create the new database environment or object before writing some code to copy the data physically into the new environment. You would also have needed to devise a process to keep the data up to date.

Are you starting to see how much easier and quicker the same process might be in Snowflake using cloning?

Which objects can be cloned

These objects can be cloned in Snowflake:

- Databases
- Schemas
- Tables
- Streams—Any unconsumed records in the stream are not available for consumption in the clone. The stream begins again at the time/point the clone was created.
- External named stages—This has no impact on the contents of the external storage location. It is simply the pointer to the storage location that is cloned.

Which objects can be cloned

• File formats

• Sequences—When a table with a column with a default sequence is cloned, the cloned table still references the original sequence object. You can instruct the table to use a new sequence by running the following command:

```
ALTER TABLE <table_name>
ALTER COLUMN <column_name> SET DEFAULT <new_sequence>.nextval;
```

• Tasks—Tasks in the clone are suspended by default, and you must execute the ALTER TASK...RESUME statement.

• Pipes—A database or schema clone includes only pipe objects that reference external stages (Amazon S3, Google Cloud Storage, or Microsoft Azure).

Which objects CANNOT be cloned

These objects cannot be cloned:

• Internal (Snowflake) stages

• Pipes—Internal (Snowflake) pipes are not cloned. A cloned pipe is paused by default.

• Views—You cannot clone a view directly. A view will be cloned if it is contained within a database or schema that you are cloning.

You must take note of which objects can and cannot be cloned. It is highly likely you will get some questions about cloning, so make sure you understand the basic concepts well.

Time Travel

Time travel allows you to query data at a point in time, within the data-retention window, using SQL. This gives you the ability to view the exact state the data was in at a specific point in time.

As with many features Snowflake provides, there's nothing you need to do in the background to maintain these copies of data. You just need to define how long to keep the version of the data, which we'll cover in the next section on data retention. By using some of the SQL extensions provided for time travel operations, you can move through versions of tables, schemas, and databases at various points in time.

You can also use cloning and time travel in tandem, which allows you to create a clone of an object by specifying an earlier point in time. You can imagine how useful this would be when, for example, you need to compare a table of data from before and after a load process.

Time Travel - Data Retention

To cater to the flexibility time travel offers, Snowflake maintains versions of data as they change over time.

Think of this as an audit trail or ledger of transactions applied to the data. Each change creates a new version of that data in the background. Snowflake will keep these versions for as long as the data-retention period is set.

In the Standard edition of Snowflake, the retention period is, by default, one day or 24 hours.

It is possible to set time travel to 0, which is equivalent to disabling it, meaning historical data is no longer available to be queried.

Time Travel - Data Retention

You must make the right choice from the outset. Extending the data-retention period from a lower to a higher number—for example, 0 to 1—doesn't mean you'll have access to that data immediately. In this instance, you'll have to wait for a day to pass until you have a full day's worth of data to access via time travel.

	Standard Edition			Enterprise Edition (and higher)		
	Min	Default	Max	Min	Default	Max
Temporary or Transient Objects	0	1	1	0	1	1
Permanent Objects	0	1	1	0	1	90

Time Travel - Data Retention

To change the data-retention period, you can use the ACCOUNTADMIN role to set the value for the DATA_RETENTION_TIMEJN_DAYS parameter.

Interestingly, you may use this parameter when creating a database, schema, or table to override the global default.

This means if you have a small amount of business-critical data in your database (and you're running the Enterprise edition or above), you could decide to set the retention period to 90 days for the objects holding that data while leaving all other objects at the default of one day.

Time Travel – Data Retention

The following code snippet shows how you can set the data-retention period at the point when you create a table and then amend the retention at a later point in time:

```
CREATE TABLE ORDERS(ORDER_ID INT, PRODUCT_ID INT,
ORDER_QUANTITY INT)
DATA_RETENTION_TIME_IN_DAYS = 60;

ALTER TABLE ORDERS SET DATA_RETENTION_TIME_IN_DAYS=30;
```

Dropping and undropping historical data

When an object is dropped and time travel is enabled, the data is not removed from the account. Instead, the version of the object is held in line with the data-retention period.

This means if someone mistakenly drops an object, you can restore it by using the UNDROP command. There's no need to concern yourself with locating and restoring backups in this situation!

Dropping and undropping historical data

When an object is dropped and time travel is enabled, the data is not removed from the account. Instead, the version of the object is held in line with the data-retention period.

This means if someone mistakenly drops an object, you can restore it by using the UNDROP command. There's no need to concern yourself with locating and restoring backups in this situation!

You can list any dropped objects using the SHOW command along with the HISTORY keyword.

Dropping and undropping historical data

This code lists the history of all tables where the name is like '%TABLE_NAME':

```
SHOW TABLES HISTORY LIKE '%TABLE_NAME';
```

This code shows the history of schemas within the Sales database:

```
SHOW SCHEMAS HISTORY IN SALES_DB;
```

This code shows the history of all databases in the Snowflake account:

```
SHOW DATABASES HISTORY;
```

Dropping and undropping historical data

Once you have found the object that you'd like to restore, you can run a statement like this one:

```
UNDROP DATABASE SALES_DB;
```

This command will restore the Sales_DB database, including all the database objects and data, exactly as it was when it was dropped.

Fail-safe

After the data-retention period associated with time travel ends, data cannot be viewed within your account.

But that's not the end of the story.

There is one final resting point for your data, which you cannot see or access directly. For a further, non-configurable 7 days after data leaves the database, data from permanent objects ends up in something called a fail-safe.

Fail-safe

If you cannot see the data in the fail-safe, then who can?

Well, only Snowflake employees can access the fail-safe, so it may take several hours to recover the data from this area.

Snowflake states that it is provided on a best-endeavor basis, meaning you should not rely on this as part of a disaster recovery scenario.

Underlying storage concepts

Storage Concepts

When you drop or truncate a table, entire copies of the data are held in time travel and fail-safe. In all other cases. Snowflake is intelligent enough to work out what data it needs to store to recover those individual rows if required. Storage usage is simply calculated as a percentage of the overall table size versus those records that have been modified.

To view the amount of storage used by fail-safe, you first must use the ACCOUNTADMIN role by default. This allows you to head into the 'Billing & Usage' tab of the Account section in Snowflake.

Recap

Domain 6.0 - Recap Part 1

- Transient and temporary tables have no fail-safe period. As a result, no additional data storage charges are incurred beyond the time travel retention period.
- By default, temporary and transient tables have a time travel retention period of one day.
- You cannot configure the fail-safe period for any object.
- Snowflake removes a lot of the administrational overhead associated with looking after a database. This includes taking backups. Snowflake handles all of this in the background as part of its continuous data lifecycle feature.

Domain 6.0 - Recap Part 1

- Fail-safe is for use only by Snowflake to recover data that may have been lost or damaged due to extreme operational failures.

- The most efficient way to create a non-production environment for testing is to create a clone of the required objects to the test environment. This process involves zero data movement and does not require any additional storage.

- You can update records in the cloned object. While this doesn't impact the source table, it does use additional storage at this point to maintain the row version.

Domain 6.0 - Recap Part 1

- Fail-safe provides a (non-configurable) 7-day period during which historical data may be recoverable by Snowflake.

- Time travel and fail-safe are the two continuous data protection features that support data recovery automatically.

Secure data shares

Secure Data Shares - Inbound

When you first set up your Snowflake account, or if you are using the trial Snowflake account, you can switch to the ACCOUNTADMIN role and head to Data > Database (in Snowsight).

By default, you'll see two databases, one called 'SNOWFLAKE' and the other called'SNOWFLAKE SAMPLE DATA'.

Secure Data Shares - Inbound

You didn't create these databases when you set up your account—so how did they get here?

Well, these databases are both inbound shares.

We can see that both these databases have the 'Source' defined as 'Share'. If you want, you can even drop these databases and get them back using data sharing.

Secure data shares - outbound

Secure Data Shares - Outbound

An outbound share is a way for you, as the data owner, to share data with other accounts.

Data Marketplace

Data Marketplace

Database objects can now be shared instantly between approved Snowflake accounts with zero latency and storage costs.

The data marketplace builds upon this core capability by opening a shared marketplace that allows the buying and selling of data.

Many businesses can improve their existing services or enhance their products by augmenting their existing data with external data. Using the data marketplace to seamlessly access new, external data sources removes the friction previously associated with this sharing of data.

Private Data Exchange

A private data exchange allows you to create a data hub for collaboration between a selected group of members who are invited. Providers can then publish data that can be discovered by consumers.

The main advantages of having a private data exchange are that it reduces data silos and allows for data collaboration at scale between internal business units or external parties such as suppliers, customers, and partners. Think of it like an exclusive version of the data marketplace for a select group of users.

A data exchange requires all data providers and data consumers to have a full Snowflake account—no reader accounts can participate in a data exchange.