WELCOME TO THE SNOWFLAKE

Ramanand Avanasiappan

2019

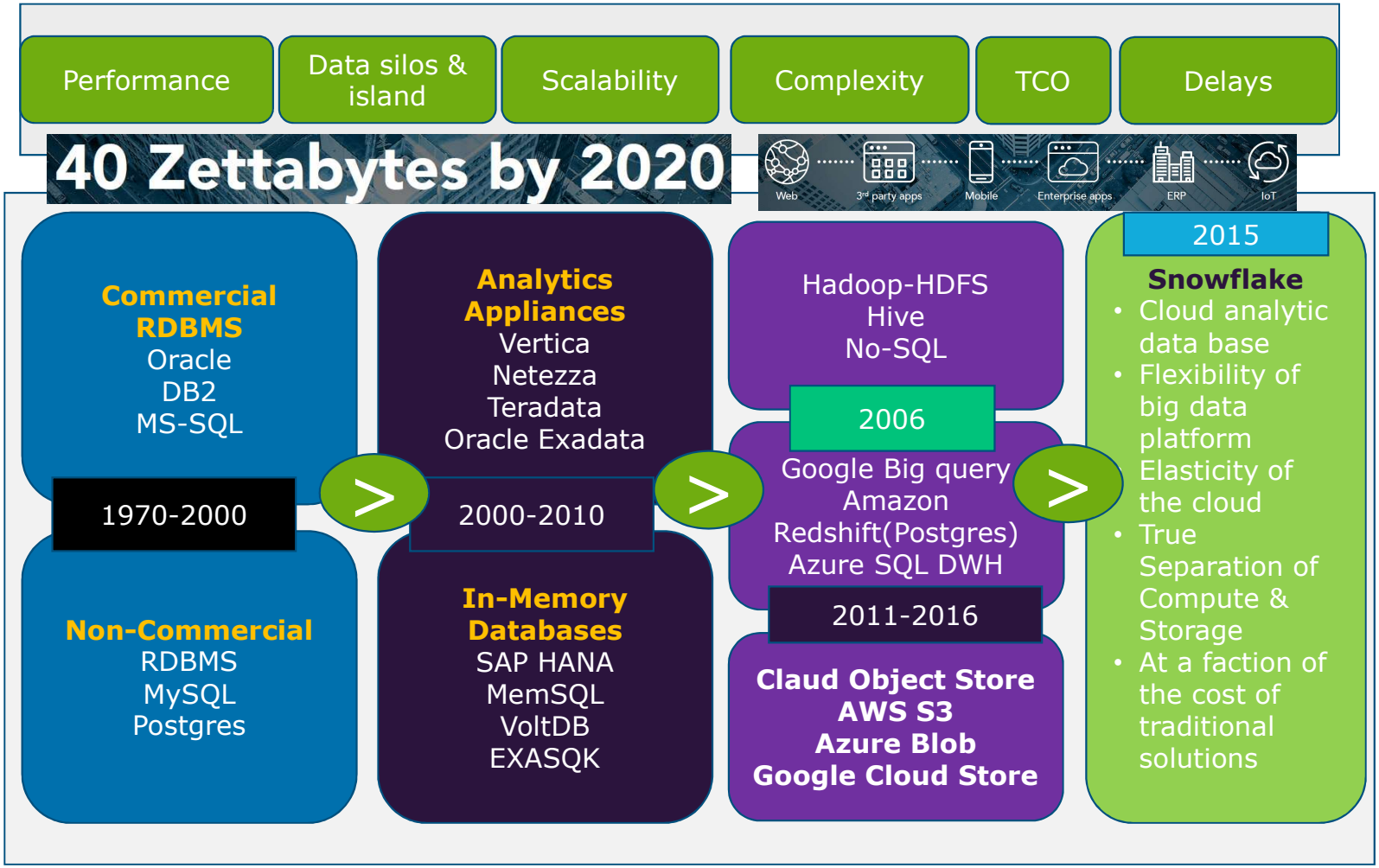
Snowflake COE I&D CDP India

Tutor Introduction

- Ramanand has over 16 years of professional experience. More than 12 years of rich experience in Teradata data warehousing implementations.
- This is his 6th year of accession with Capgemini India played various roles as Teradata Architect, Lead Data modeler, Solution architect, Lead Data Architect and Delivery manager for the larger I&D engagements.
- Recent 1 year he is handling India Snowflake Practice supporting proposals, Solution & implementation, Accelerators, Competency built and Trainings.
- Ramanand has completed M.Sc. Computer Technology from Bharathiar University Coimbatore Tamil Nadu. In his prior experience he has worked with Cognizant, Wipro, Mphasis and Target Corporation.
- In his current role, Ramanand is part of Cloud Data Platform and predominantly continue focusing on Snowflake practice and resident solution architect for Snowflake CDP engagements.

snowflake
YOUR DATA, NO LIMITS

Today's Data Platform Challenges and Options



Agenda

- **Day 1: Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview**
  - ✓ **Snowflake User Interface**
  - ✓ **Snowflake Administration**
  - ✓ **Snowflake Security features**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Day 2: Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Day 1: Get started with Snowflake

Company & Product

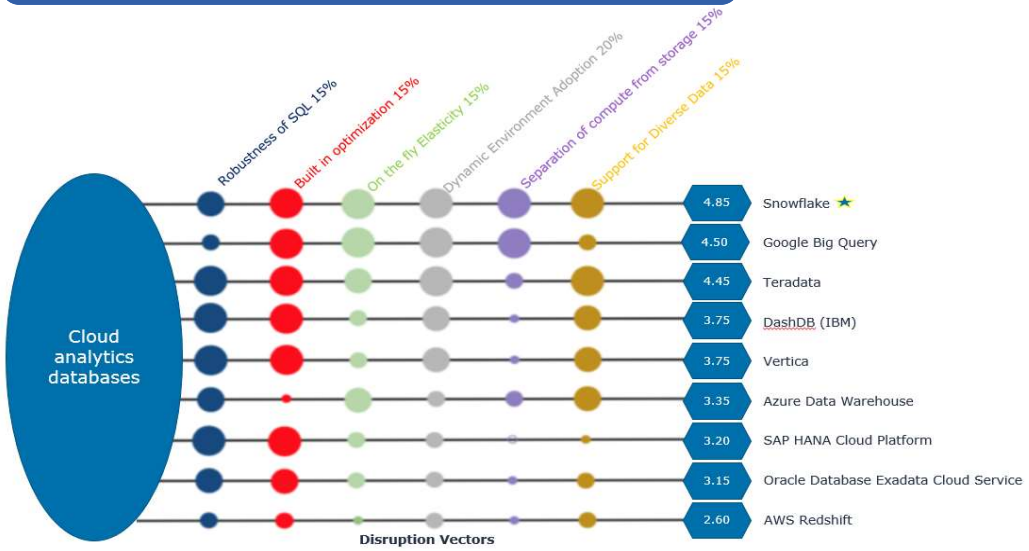
Snowflake Computing was founded in 2012

It came out of stealth mode in October 2014

It became generally available in June 2015

- Snowflake Computing sells a cloud-based data storage and analytics service called Snowflake Elastic Data Warehouse.
- It allows corporate users to store and analyze data using cloud-based hardware and software. The data is stored in Amazon S3. According to the CEO, the software relies on the public cloud ecosystem, and isn't using Hadoop.
- The developer was identified as a "Cool Vendor" in Gartner's Magic Quadrant and won first place at the 2015 Strata + Hadoop World startup competition. It became generally available in June 2015 and had 80 organizations using it at that time.
- 2017 **Snowflake** Ranked as the **#1 cloud data warehouse** by industry analyst firm **Gigaom** Research For more details
- As of January **2018**, Snowflake is used by over **1000** organizations, including Capital One, Rent the Runway and Adobe.

Cloud analytics databases Comparison



*2017 Snowflake Ranked as the #1 cloud data warehouse by industry analyst firm Gigaom Research For more details https://resources.snowflake.net/modernize-your-data-warehouse/gigaom-sector-roadmap-cloud-analytic-databases-2017

What is Snowflake

Vision

Provide all users anytime, anywhere insights so they can make actionable decisions based on data

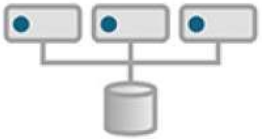
Solution

Next-generation data warehouse built from the ground up for the cloud and for today's data and analytics

- Snowflake is the only data warehouse built for the cloud
- The Data Warehouse as a Service (DWaaS) supports
  - Demanding
  - High performance architectures
  - A broad variety of data types
  - It supports to analytics, enterprise reporting, data sharing, or predictive analysis
- Snowflake is the cloud-native database that makes it all accessible.

Traditional versus Cloud-Built Architecture

Traditional architectures



Shared-disk	Shared-nothing
Shared storage	Decentralized, local storage
Single cluster	Single cluster

Built-for-the-cloud architecture

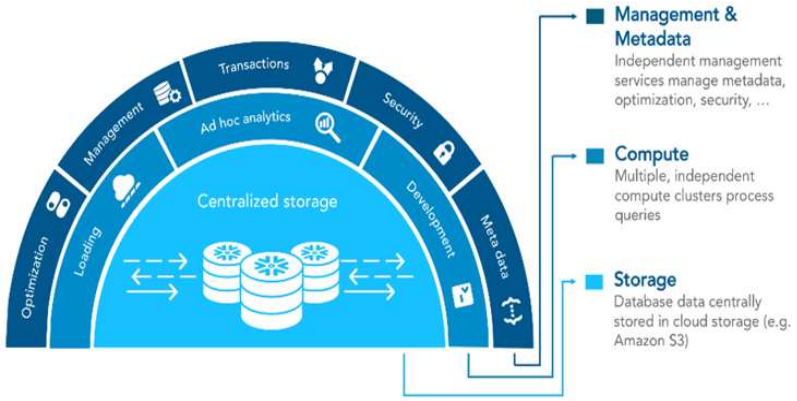
Multi-cluster, shared data
Centralized, scale-out storage
Multiple, independent compute clusters

Key Concepts & Architecture:

Snowflake is an analytic data warehouse provided as Software-as-a-Service (SaaS). Snowflake provides a data warehouse that is faster, easier to use, and far more flexible than traditional data warehouse offerings. Snowflake's data warehouse is not built on an existing database or "big data" software platform such as Hadoop.

The Snowflake data warehouse uses a new SQL database engine with a unique architecture designed for the cloud. To the user, Snowflake has many similarities to other enterprise data warehouses, but also has additional functionality and unique capabilities.

- ❑ Data Warehouse as a Cloud Service
- ❑ Snowflake Architecture
  - ❑ Database Storage
  - ❑ Query Processing
  - ❑ Cloud Services
- ❑ Connecting to Snowflake



Snowflake's data warehouse is a true **SaaS** offering. More specifically:

- There is **no hardware** (virtual or physical) for you to select, install, configure, or manage.
- There is **no software** for you to install, configure, or manage.
- Ongoing maintenance, management, and tuning is handled by Snowflake.

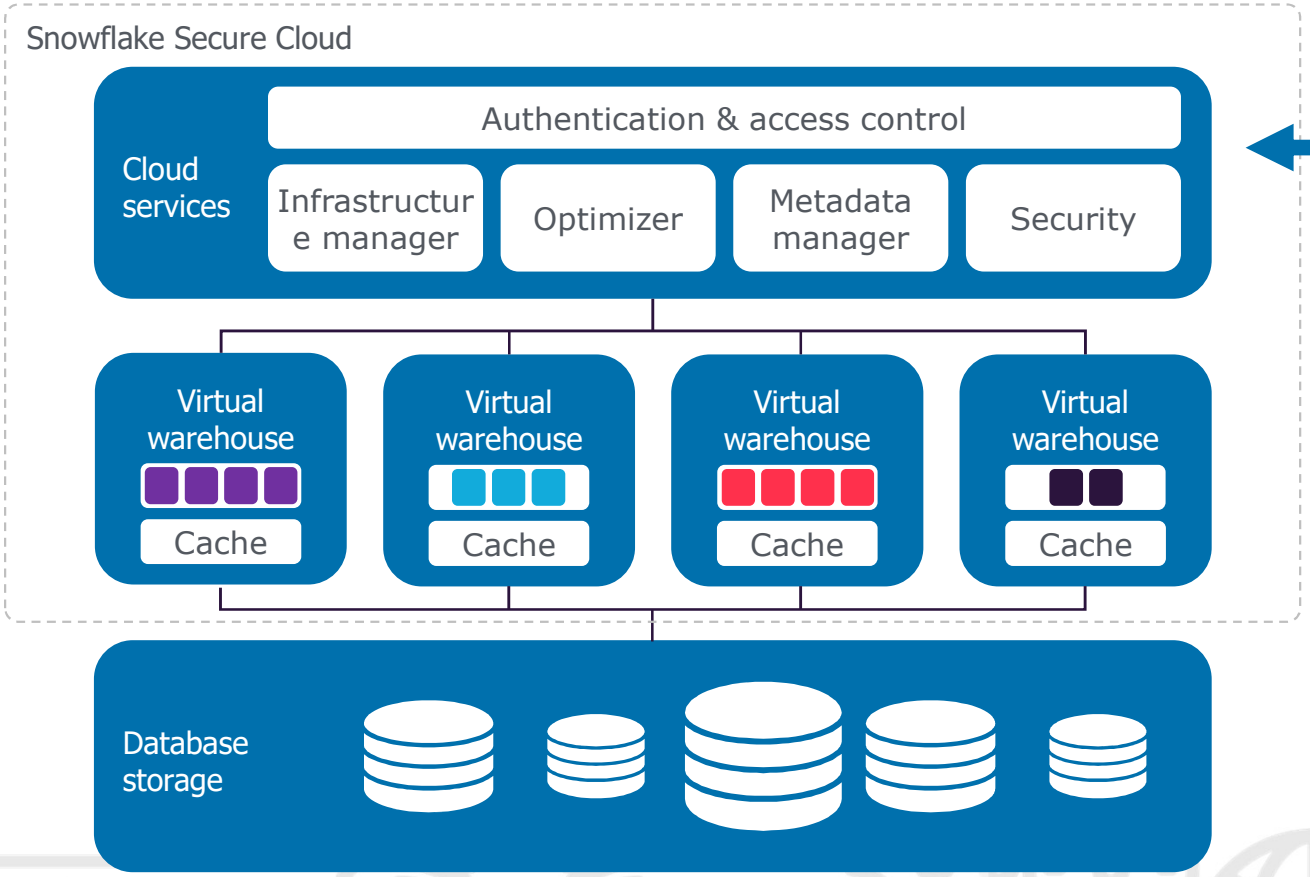
Snowflake **runs completely on cloud infrastructure**. All components of Snowflake's service (other than an optional command line client), run in a public cloud infrastructure.

Snowflake uses virtual compute instances for its compute needs and a storage service for persistent storage of data.

Snowflake cannot be run on private cloud infrastructures (on-premises or hosted).

Snowflake is not a packaged software offering that can be installed by a user. Snowflake manages all aspects of software installation and updates.

A deeper look…



JDBC / ODBC /Python

- Storage decoupled from compute
- All data in one place
- Dynamically combine storage and compute

Data Warehouse as a Cloud Service

- Snowflake provides true SAS features with
  - No hardware (virtual or physical to select, install, configure, or manage.
  - No software for you to install, configure, or manage.
  - No Ongoing maintenance, management, and tuning since it is handled by Snowflake.

- It Runs completely on cloud infrastructure

- All Services (except command line interface) run in a public cloud infrastructure.

- It uses virtual compute instances for its compute needs and a storage service for persistent storage of data

- It is not a packaged software offering that can be installed by a user

- Snowflake manages all aspects of software installation and updates.

Cloud Platforms

- Software-as-a-Service (SaaS) that runs completely on cloud infrastructure
- All Storage, compute, and services can run on all different cloud vendors like AWS, Google and Azure
- Snowflake Services can be hosted on AWS, Azure and GCP.
- Choice of selection of Vendor is completely independent from snowflake account

- Following criteria to be considered while selecting cloud services
  - **Billing** - Differences in unit costs for credits and data storage are calculated by geographical region and not by cloud platform.
  - **Data Loading** – supports loading data from files staged in any of the locations such as Internal (i.e. Snowflake) stages, Amazon S3 and/or Microsoft Azure Blob storage.
    - Snowflake supports bulk data loading and continuous data loading (Snowpipe)

Snowflake Editions

- Snowflake Provides following Editions –
  - Standard Edition
  - Premier Edition
  - Enterprise Edition
  - Enterprise Edition for Sensitive Data (ESD)
  - Virtual Private Snowflake (VPS)

Standard
- Business hour support M-F
- 24 hours of time travel
- Always-on Enterprise Grade encryption in transit and at rest
- Zero copy clone

Premier

STANDARD +
- Premier Support: 24 x 365
- Faster support response time
- SLA w/refund for outage

Enterprise

PREMIER +
- Multi-Cluster warehouse
- Up to 90 days of time travel
- Federated authentication
- Annual rekey of all encrypted data

Enterprise for Sensitive Data

ENTERPRISE +
- Support for HIPAA
- Data encryption everywhere
- Virtual warehouses run on dedicated servers
- Enhanced security policy
- Optional support for customer managed keys

Standard Edition

- Providing full unlimited access to all of Snowflake's standard features.
- provides a strong balance between features, level of support, and cost.
- Features are supported under various areas
  - **Security and Data Protection**
    - SOC 2 Type II certification.
    - Network policies for limiting/controlling site access by user IP address.
    - Automatic encryption of all data.
    - Support for multi-factor authentication.
    - Object-level access control.
    - Access to all modified and deleted data (up to 1 day) through Time Travel.
    - Disaster recovery of modified and deleted data (up to 7 days beyond Time Travel) through Fail-safe.
  - **SQL**
    - Standard SQL, including most DDL and DML defined in SQL:1999.
    - Advanced DML such as multi-table INSERT, MERGE, and multi-merge.
    - Broad data type support.
    - Multi-statement transactions.
    - User-defined functions (UDFs) with support for both SQL and JavaScript.
    - Native support for semi-structured data (JSON, Avro, ORC, Parquet)

Standard Edition (Cont…)

- **Import and Export**
  - Support for bulk loading from delimited flat files (CSV, TSV, etc.) and semi -structured data files (JSON, Avro, ORC, Parquet, and XML).
  - Support for bulk unloading to delimited flat files and JSON files.

- **Tools and Interfaces**
  - Full-featured web-based interface and command line client.
  - Programmatic interfaces for Python, Node.js, and Spark.
  - Native support for JDBC and ODBC.
  - Extensive ecosystem for connecting to ETL, BI, and other third-party systems.

Browser-based web interface

SnowSQL, the Snowflake command line client

Any client application connected via JDBC or ODBC

Premier Edition

- In Addition to Standard features snowflake provides following additional features

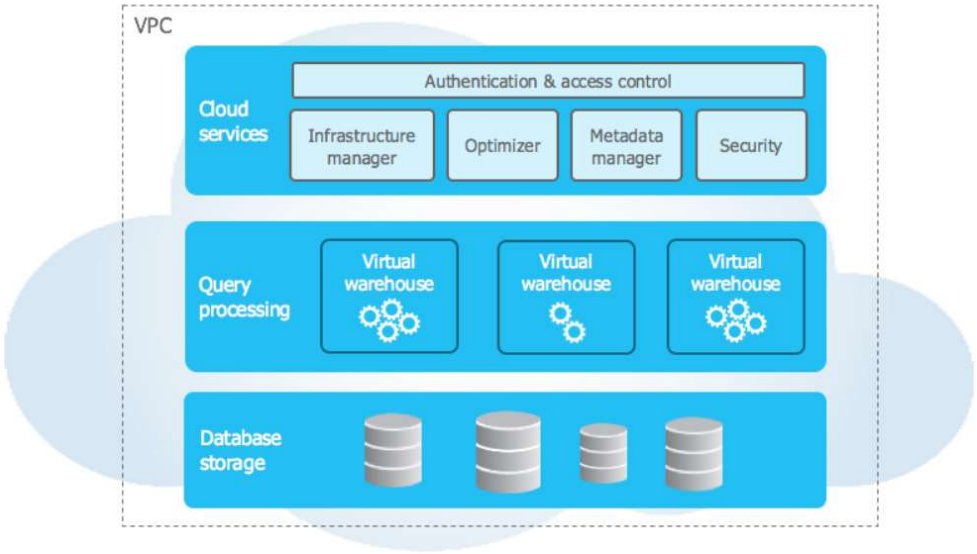
- **Security and Data Protection**
  - Federated authentication and SSO (Single Sign On) for centralizing and streamlining user authentication.
  - Periodic rekeying of encrypted data for increased protection.
  - Extended data retention for Time Travel (up to 90 days).
- **Automated Resource Management**
  - Multi-cluster warehouses for scaling compute resources to meet fluctuating concurrency needs.

Enterprise Edition

- Enterprise Edition for Sensitive Data (ESD)

Key Concepts & Architecture:

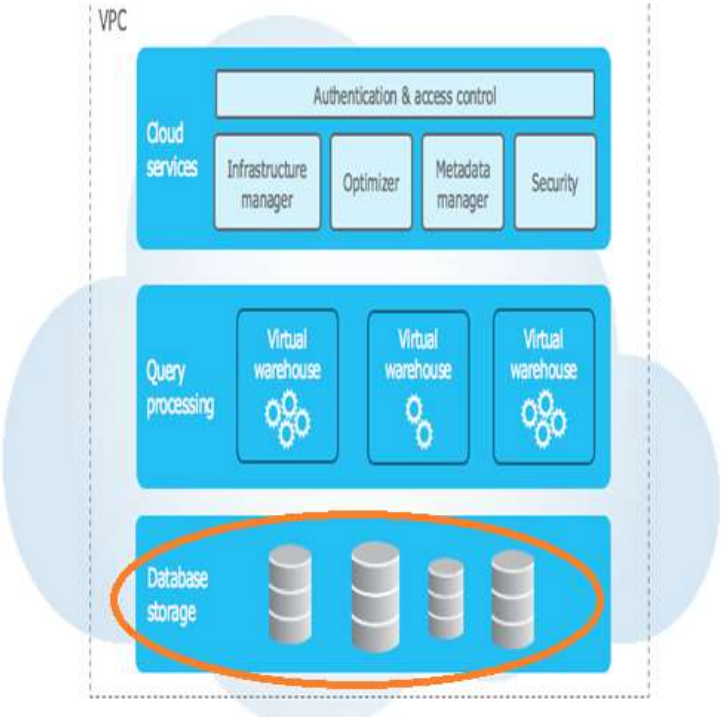
- Snowflake Architecture
  - Database Storage
  - Query Processing
  - Cloud Services
- Connecting to Snowflake



- It is hybrid of traditional shared-disk database architectures and shared-nothing database architectures.
- uses a central data repository for persisted data that is accessible from all compute nodes in the data warehouse
- processes queries using MPP (massively parallel processing) compute clusters
- Leads to performance and scale-out benefits of a shared-nothing architecture.
- Provides 3 keys unique key layers
  - Data base storage
  - Query Processing
  - Cloud Services

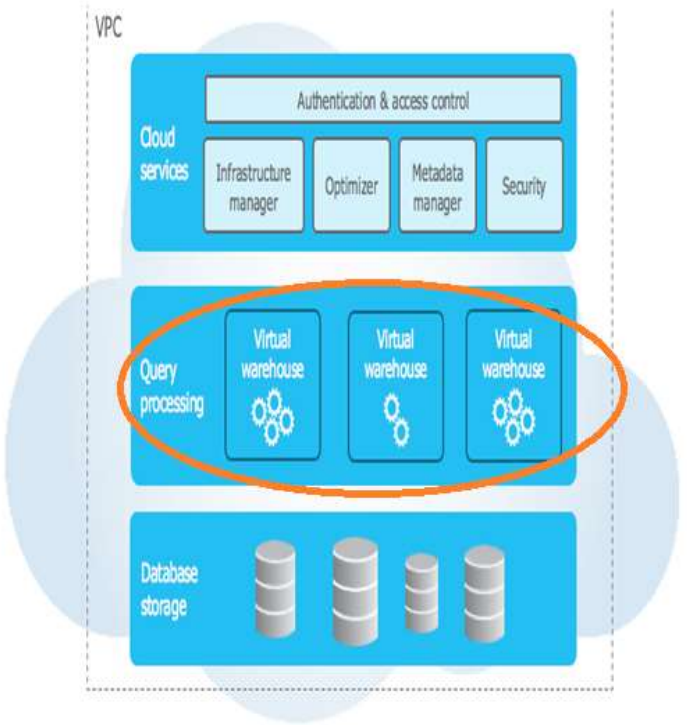
Database Storage

- Snowflake reorganizes that data into its internal optimized, compressed, columnar format.
- Snowflake manages as aspects of storage that includes
  - the organization,
  - file size,
  - structure,
  - compression,
  - metadata,
  - statistics,
  - and other aspects of data storage
- Data in snowflakes is not directly visible nor accessible by customers
- Data can be accessed through SQL



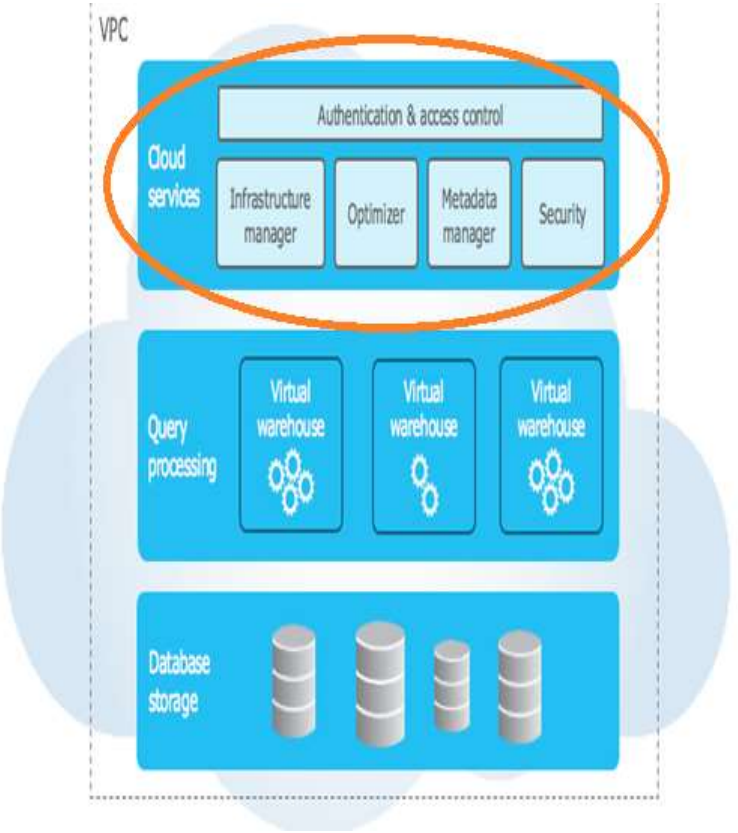
Query Processing

- Query execution is performed in the processing layer.
- Processes queries using “virtual warehouses”.
- Virtual warehouse is an MPP compute cluster
- Virtual warehouse is an independent compute cluster that does not share compute resources with other virtual warehouses.
- Benefit
  - Virtual warehouse has no impact on the performance of other virtual warehouses.

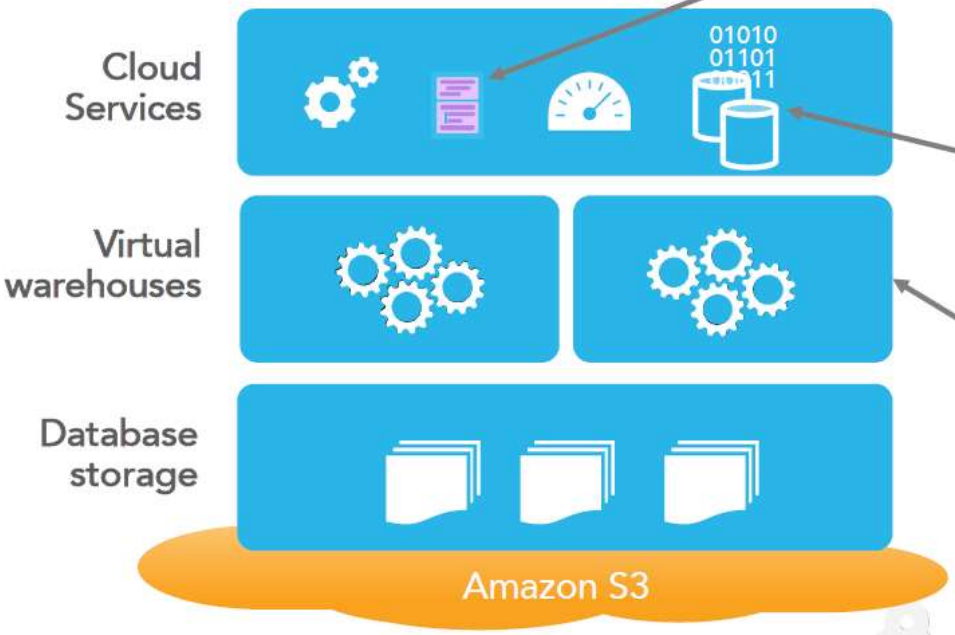


Cloud Services

- The cloud services layer is a collection of services that coordinate activities across Snowflake.
- These services tie together all of the different components of Snowflake in order to process user requests, from login to query dispatch.
- The cloud services layer also runs on compute instances
- The Services includes
  - Authentication
  - Infrastructure management
  - Metadata management
  - Query parsing and optimization
  - Access control



Dynamic caching for optimal query performance



Result Cache

- All query results stored for 24 hours unless underlying data changes
- Identical queries are returned instantly without requiring compute

Metadata Cache

- Improves compile times for queries against commonly used tables

Virtual Warehouse Cache

- Data loaded into warehouses is stored in local SSD storage
- Cache entries are invalidated if underlying data changes

The Snowflake Elastic Data Warehouse

All-new SQL data warehouse

No legacy code or constraints

Designed for the cloud

Running in Amazon Web Services

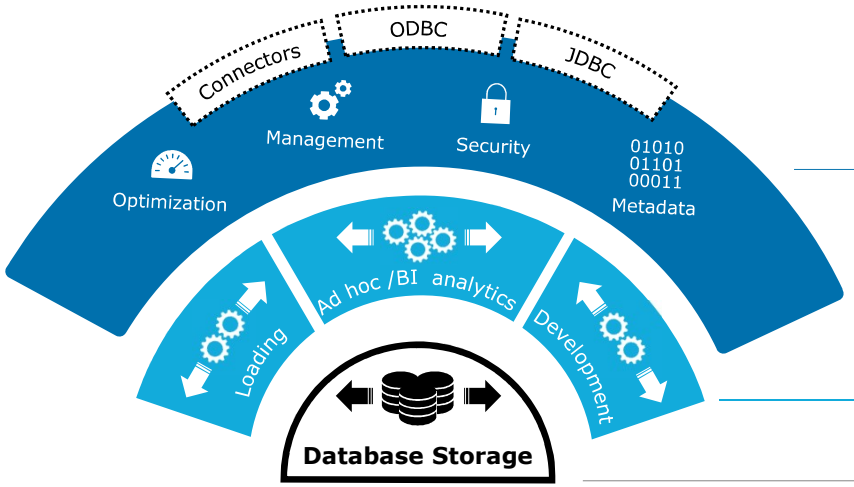
Delivered as a service

No infrastructure, knobs or tuning to manage

All your data

Deploy Structured and Semi-structured data in one place

Enabling Strong Concurrency to allow Analytics at Scale



Enabling Concurrency

Concurrency through scaling and warehouse/storage separation

Single service

Scalable, resilient cloud services layer coordinates access & management

Elastically scalable compute

Multiple "virtual warehouse" compute clusters scale horsepower & concurrency

Centralized storage

Instant, automatic scalability & elasticity

No infrastructure, knobs, or tuning

Infrastructure Management

Hardware, software, availability, resiliency, disaster recovery managed by Snowflake

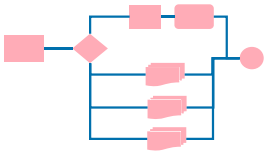
Data Storage Management

Adaptive data distribution, automatic compression, automatic optimization

Metadata Management

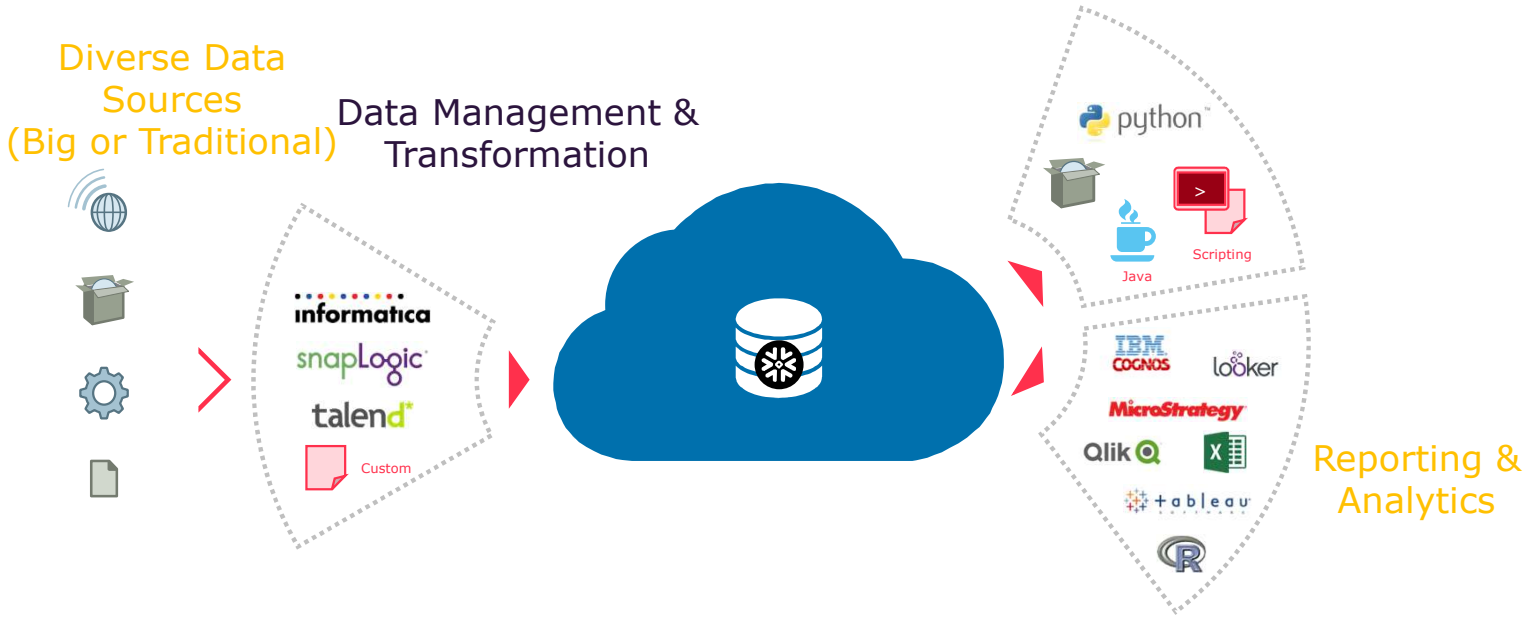
Automatic statistics collection, scaling, and redundancy

Manual Query Optimization



Dynamic optimization, parallelization, and concurrency management

Fits with your Ecosystem



Getting Started

Before You Begin

Snowflake can be accessed from any of the following:
•Browser-based web interface
•SnowSQL, the Snowflake command line client
•Any client application connected via JDBC or ODBC

SnowSQL OS Platform Requirements

SnowSQL can be installed on the following platforms:
•Redhat-compatible Linux operating systems
•Mac OS (64-bit)
•Windows (64-bit)
Other platforms have not been tested at this time and may not be compatible with SnowSQL. For example, some Linux variants may not have the libraries that the SnowSQL client needs by default.

SnowSQL is available for download from the Snowflake web interface

Browser Requirements

You can use any of the following browsers to access the Snowflake web interface:

Supported Browser	Minimum Version
Chrome	47
Safari	9
Firefox	45
Internet Explorer	11
Opera	36
Edge	12

We recommend using Google Chrome.

JDBC and ODBC OS Platform Requirements

Both drivers can be installed on the following platforms:
•Linux
•Mac OS
•Windows:
    •64-bit for the JDBC driver
    •32-bit and 64-bit for the ODBC driver
Other platforms have not been tested at this time and may not be compatible with the drivers.
The JDBC driver is available for download from the Maven Central Repository: https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/

The ODBC driver is available for download from the Snowflake web interface

Key Concepts & Architecture:

❑ Connecting to Snowflake

Connecting to Snowflake (Interface):

Snowflake supports multiple ways of connecting to the service:

•A web-based user interface from which all aspects of managing and using Snowflake can be accessed.

•Command line clients (e.g. SnowSQL) which can also access all aspects of managing and using Snowflake.

•ODBC and JDBC drivers that can be used by other applications (e.g. Tableau) to connect to Snowflake.

•Native connectors (e.g. Python) that can be used to develop applications for connecting to Snowflake.

•Third-party connectors that can be used to connect applications such as ETL tools (e.g. Informatica) and BI tools to Snowflake.

Recap

Snowflake is…
- An all-new data warehouse
- Designed for the cloud
- Combined structured and semi-structured data in an optimized manner

Snowflake delivers…
- One place for diverse data
- Easier, faster analytics
- Elastic scaling for any scale of data, workload, & concurrency
- Without the cost and complexity of alternatives

Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface**
  - ✓ **Snowflake Administration**
  - ✓ **Snowflake Security features**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Day 1: Get started with Snowflake

User Interface Walk Through
Snowflake General Reference (SQL)

- ✓ **Snowflake User Interface**
  - User Interface Walk Through
  - Snowflake General Reference (SQL)
  - User Interface Quick Tour – [Demo] [https://www.youtube.com/watch?v=8vDqf1Nd0Hk&feature=youtu.be]

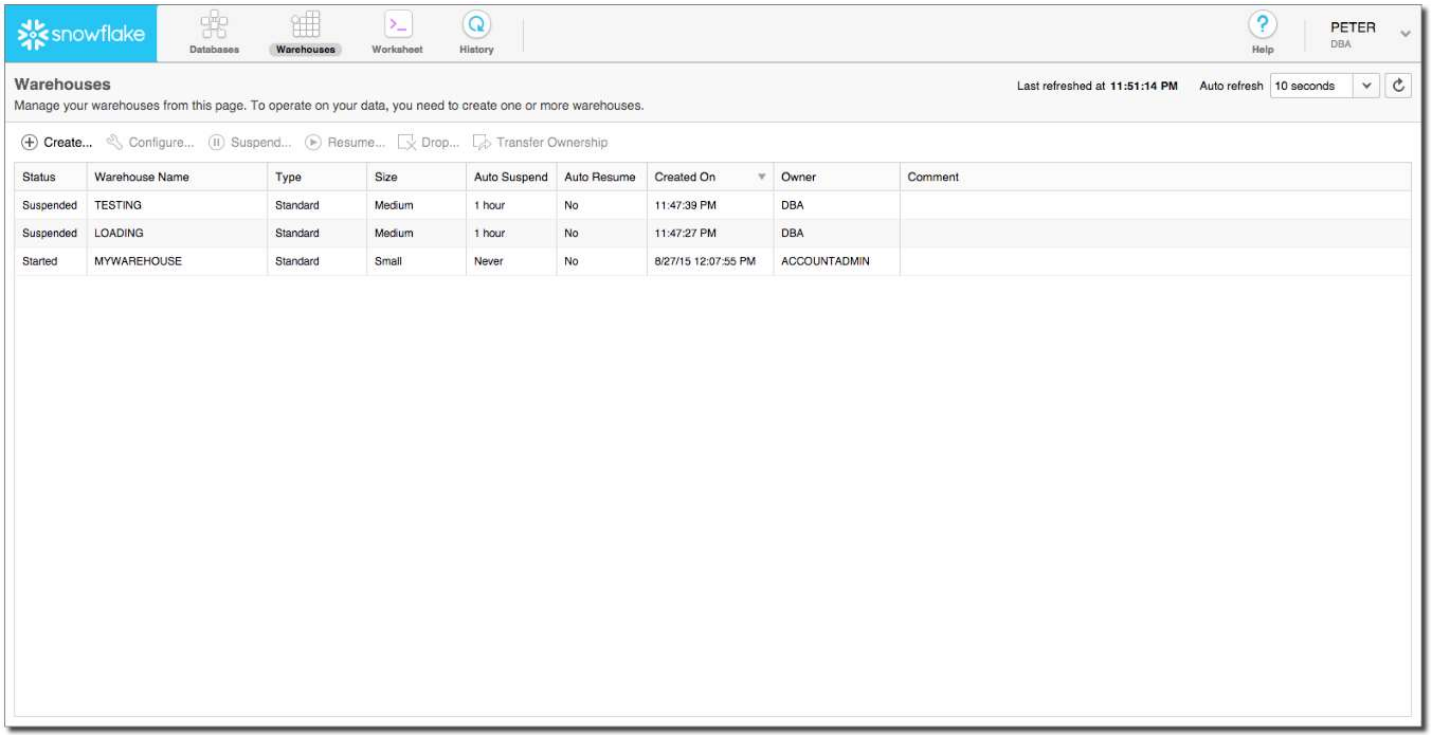
Web Interface

- Snowflake web-based graphical interface, you can create and manage all Snowflake objects, including virtual warehouses, databases, and all database objects.
- Can also use the interface to load limited amounts of data into tables, execute ad hoc queries and perform other DML/DDL operations, and view past queries.
- And the interface is where you can change your Snowflake user password and specify other preferences, such as your email address.
- In addition, if you have the required administrator roles, you can perform administrative tasks in the interface, such as creating and managing users. For more information about the administrative tasks you can perform

Below are the interfaces:

•Warehouses Page
•Databases Page
•Worksheet Page
•History Page
•Help Menu
•User Preferences Menu

Warehouses Page

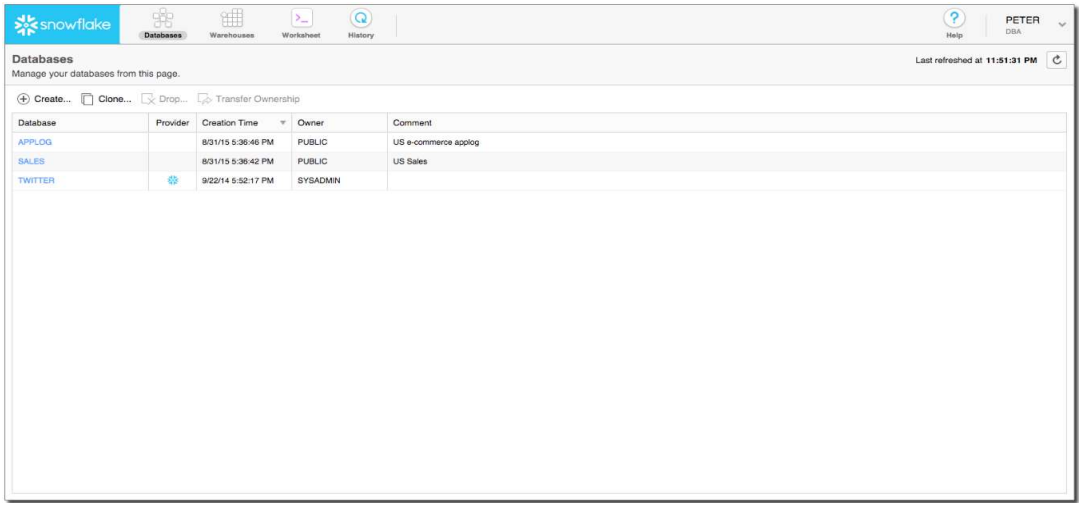


Status	Warehouse Name	Type	Size	Auto Suspend	Auto Resume	Created On	Owner	Comment
Suspended	TESTING	Standard	Medium	1 hour	No	11:47:39 PM	DBA	
Suspended	LOADING	Standard	Medium	1 hour	No	11:47:27 PM	DBA	
Started	MYWAREHOUSE	Standard	Small	Never	No	8/27/15 12:07:55 PM	ACCOUNTADMIN	

Tasks you can perform in this page (if you have the necessary privileges) include:

- Create or drop a warehouse.
- Suspend or resume a warehouse.
- Configure a warehouse.
- Transfer ownership of a warehouse to a different role.

Databases Page



This picture shows information about the databases you have created or have privileges to access. Tasks you can perform in this page (if you have the necessary privileges) include:

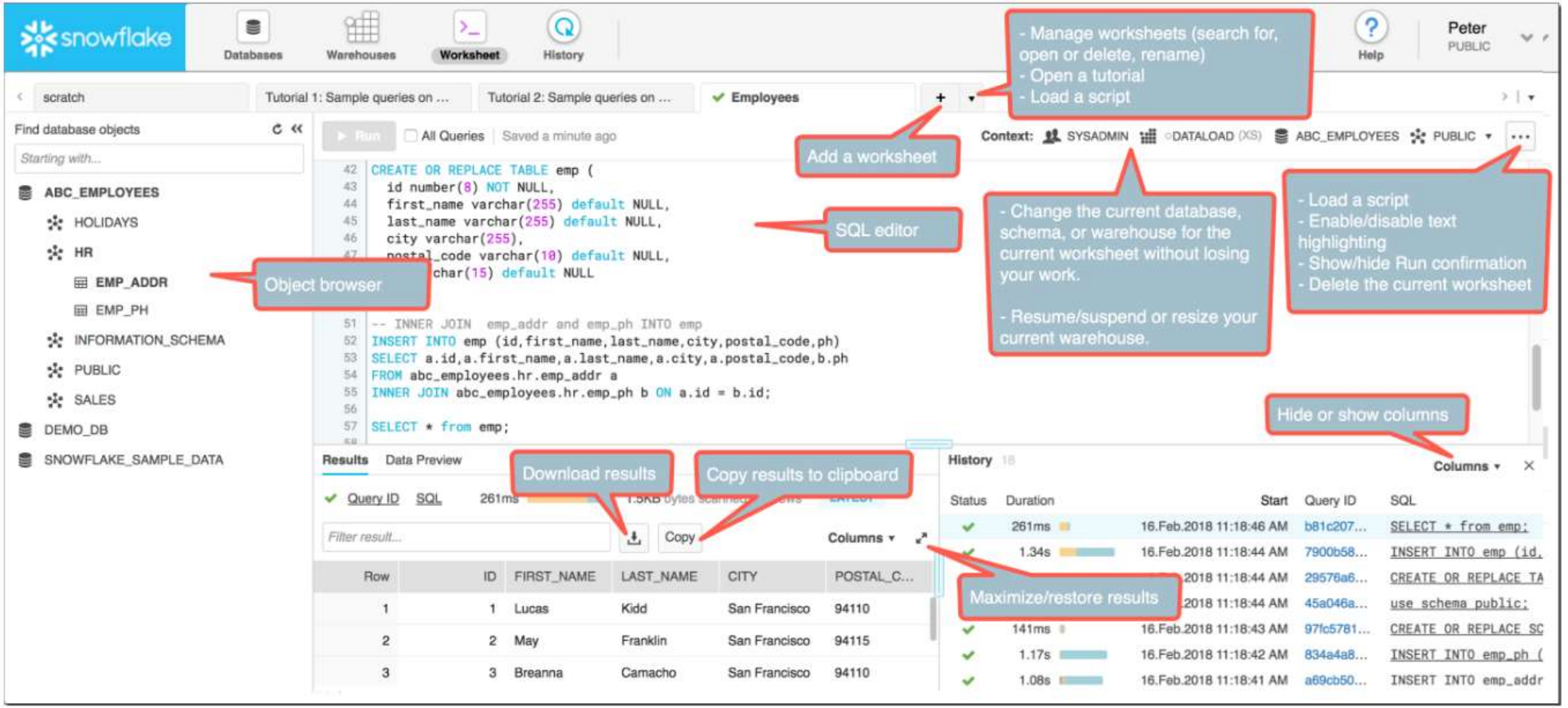
- Create, clone, or drop a database.
- Transfer ownership of a database to a different role.

In addition, you can click the name of a database to view and perform tasks on the objects in the database:

	Create	Clone	Drop	Modify	Load Data	Transfer Ownership
Tables	✓	✓	✓		✓	✓
Views	✓		✓			✓
Schemas	✓	✓	✓			✓
Stages	✓	✓	✓	✓		✓
File Formats	✓	✓	✓	✓		✓
Sequences	✓	✓	✓	✓		✓

User Interface Walk Through
Snowflake General Reference (SQL)

Worksheet Page



Worksheet Page

This page provides a powerful interface for entering and submitting SQL queries, as well as performing DDL and DML operations, and viewing the results side-by-side as your queries/operations complete. Tasks you can perform in this page include:

•Run ad hoc queries and other DDL/DML operations in a worksheet, or load SQL script files.

•Open concurrent worksheets, each with its own separate session.

•Save and reopen worksheets.

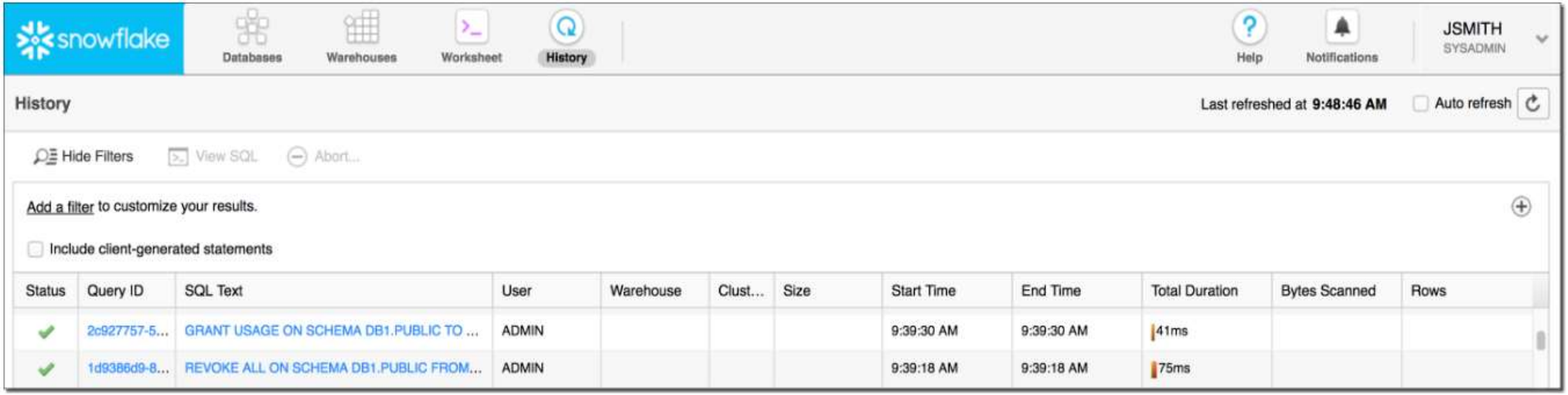
•Log out of Snowflake or switch roles within a worksheet, as well as refresh your browser, without losing your work:

- If you log out of Snowflake, any active queries stop running.
- If you're in the middle of running queries when you refresh, they will resume running when the refresh is completed.

•Resize the current warehouse to increase or decrease the compute resources utilized for executing your queries and DML statements.

•Export the result for a selected statement (if the result is still available).

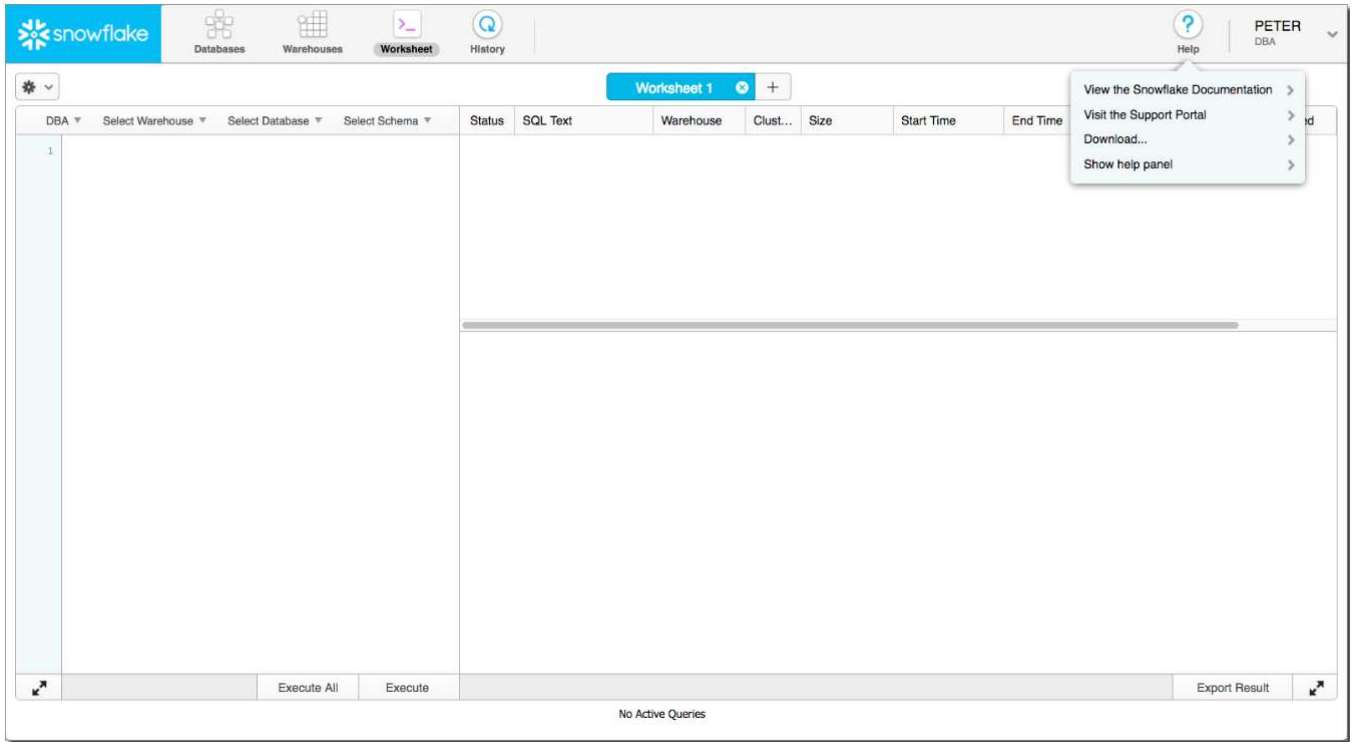
History Page



This page allows you to view and drill into the details of all queries executed in the **last 14 days**. The page displays a historical listing of queries, including queries executed from SnowSQL or other SQL clients. You can perform the following tasks on this page:

- Filter queries displayed on the page.
- Scroll through the list of displayed queries. The list includes (up to) **100 queries**. At the bottom of the list, if more queries are available, you can continue searching.
- Abort a query that has not completed yet.
- View the details for a query, including the result of the query. Query results are available for a 24-hour period. This limit is not adjustable.
- Change the displayed columns, such as status, SQL text, ID, warehouse, and start and end time, by clicking any of the column headers.

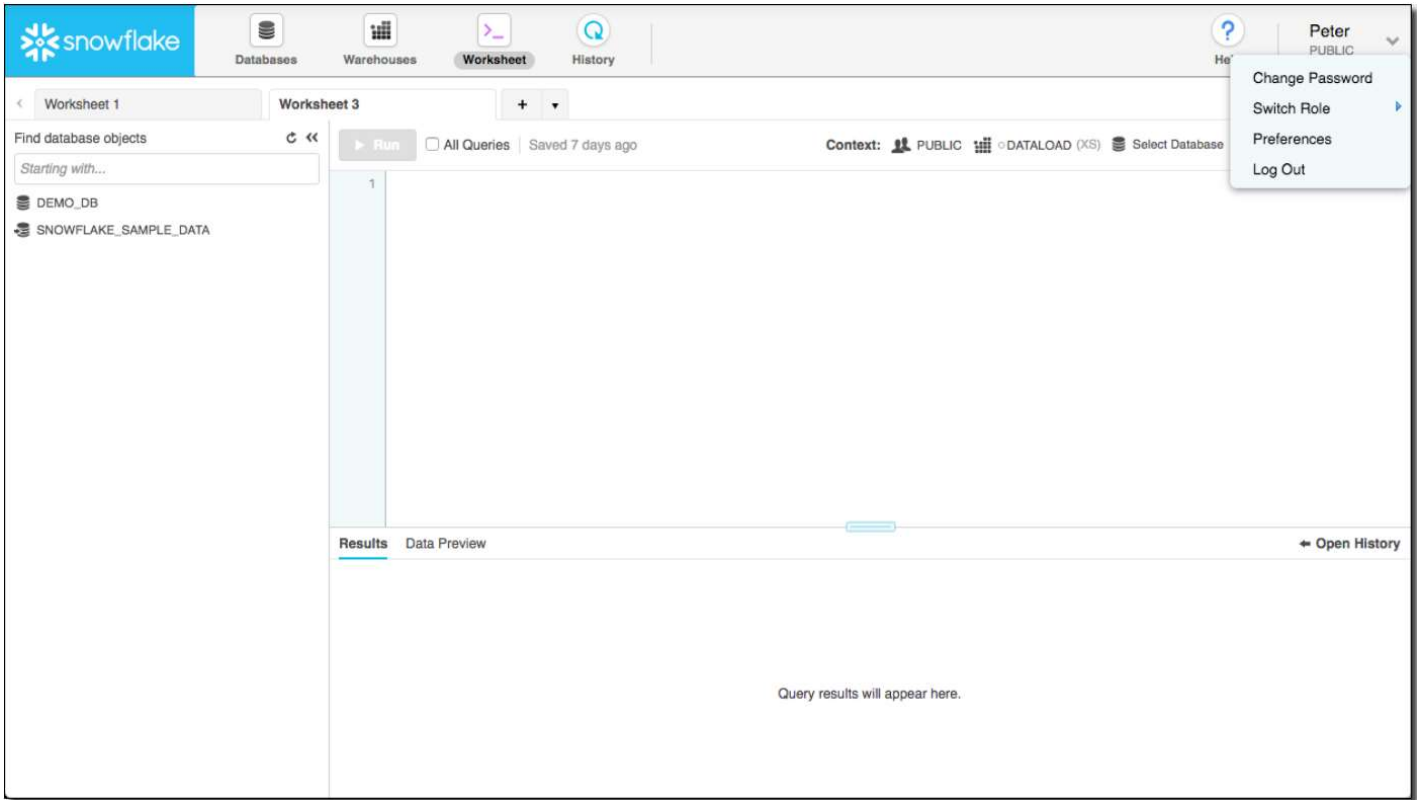
Help Menu



From the dropdown menu, choose one of the following actions:

- **View the Snowflake Documentation** in a new browser tab/window.
- **Visit the Support Portal** in a new browser tab/window.
- **Download…** the Snowflake clients by opening a dialog box where you can:
  - Download the Snowflake CLI client (SnowSQL) and ODBC driver.
  - View download info for the Snowflake JDBC driver, Python components, Node.js driver, and Snowflake Connector for Spark.
- **Show help panel** with context-sensitive help for the current page.

User Preferences Menu



You can then change your password or security role for the session (if you have multiple roles assigned to you). For more information about security roles and how they influence the objects you can see in the interface and the tasks you can perform.
You can also use this dropdown to:
•Set your email address for notifications (if you are an account administrator).
•Close your current session and exit the Snowflake web interface.

Important:

When you exit the interface, Snowflake cancels any queries that you submitted during this session and are still running.

➢ Snowflake General Reference (SQL)

•Data Types —Snowflake supports most basic SQL data types (with some restrictions) for use in columns, local variables, expressions, parameters, and any other appropriate/suitable locations. Data types are automatically coerced whenever necessary and possible. supported data types (VARCHAR, NUMBER, DATE, etc.) in Snowflake.

Summary of Data Types

Provides general reference information for using Snowflake:

•Data Types —Snowflake supports most basic SQL data types (with some restrictions) for use in columns, local variables, expressions, parameters, and any other appropriate/suitable locations. Data types are automatically coerced whenever necessary and possible. supported data types (VARCHAR, NUMBER, DATE, etc.) in Snowflake:

- •Summary of Data Types
- •Numeric Data Types
- •String & Binary Data Types
- •Logical Data Types
- •Date & Time Data Types
- •Semi-structured Data Types
- •Unsupported Data Types

Summary of Data Types

Snowflake supports most SQL data types:

Category	Type	Notes
Numeric Data Types	NUMBER	Default precision and scale are (38,0).
	DECIMAL	Synonymous with NUMBER.
	NUMERIC	Synonymous with NUMBER.
	INT, INTEGER, BIGINT, SMALLINT	Synonymous with NUMBER except precision and scale cannot be specified.
	FLOAT, FLOAT4, FLOAT8 [1]	
	DOUBLE [1]	Synonymous with FLOAT.
	DOUBLE PRECISION [1]	Synonymous with FLOAT.
	REAL [1]	Synonymous with FLOAT.
String & Binary Data Types	VARCHAR	Default (and maximum) is 16,777,216 bytes.
	CHAR, CHARACTER	Synonymous with VARCHAR except default length is VARCHAR(1).
	STRING	Synonymous with VARCHAR.
	TEXT	Synonymous with VARCHAR.
	BINARY	
	VARBINARY	Synonymous with BINARY.
Logical Data Types	BOOLEAN	Currently only supported for accounts provisioned after January 25, 2016.
Date & Time Data Types	DATE	
	DATETIME	Alias for TIMESTAMP_NTZ
	TIME	
	TIMESTAMP	Alias for one of the TIMESTAMP variations (TIMESTAMP_NTZ by default).
	TIMESTAMP_LTZ	TIMESTAMP with local time zone; time zone, if provided, is not stored.
	TIMESTAMP_NTZ	TIMESTAMP with no time zone; time zone, if provided, is not stored.
	TIMESTAMP_TZ	TIMESTAMP with time zone.
Semi-structured Data Types	VARIANT	
	OBJECT	
	ARRAY	

[1] A known issue in Snowflake displays FLOAT, FLOAT4, FLOAT8, REAL, DOUBLE, and DOUBLE PRECISION as FLOAT even though they are stored as DOUBLE.

Numeric Data Types

This topic describes the numeric data types supported in Snowflake, along with the supported formats for numeric constants/literals.

In this Topic:

Data Types for Fixed-point Numbers

Snowflake supports the following data types for fixed-point numbers.

NUMBER

Numbers up to 38 digits, with a specified precision and scale:

Precision:	Total number of digits allowed.
Scale:	Number of digits allowed to the right of the decimal point.

By default, precision is 38 and scale is 0, i.e. NUMBER(38, 0). Note that precision limits the range of values that can be inserted into (or cast to) columns of a given type. For example, the value 999 fits into NUMBER(38,0) but not into NUMBER(2,0).

DECIMAL , NUMERIC
Synonymous with NUMBER.

INT , INTEGER , BIGINT , SMALLINT , TINYINT , BYTEINT
All integer data types are synonymous with NUMBER, except that precision and scale cannot be specified, i.e. always defaults to NUMBER(38, 0).

Impact of Precision and Scale on Storage Size

Precision (total number of digits) does not impact storage, i.e. the storage requirements for the same values in columns with different precisions, such as NUMBER(2,0) and NUMBER(38,0), are the same. For each micro-partition, Snowflake determines the minimum and maximum values for a given column and uses that range to store all values for that column in the partition.
For example:

- If a column contains 5 values (e.g, 0,1,2,3,4), each of the 5 values consumes 1 byte (uncompressed; actual storage size is reduced due to compression).
- If a column contains longer values (e.g., 0,1,2,3,4,10000000), each of the values consumes 4 bytes (uncompressed).

However, scale (number of digits following the decimal point) does have an impact on storage. For example, the same value stored in a column of type NUMBER(10,5) consumes more space than NUMBER(5,0). Also, processing values with a larger scale could be slightly slower and consume more memory

Fixed-point Examples in Table Columns

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

desc table test_fixed;
```

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


Data Types for Floating Point Numbers

Snowflake supports the following data types for floating point numbers.

FLOAT , FLOAT4 , FLOAT8
Snowflake uses double-precision (64 bit) IEEE 754 floating point numbers.

DOUBLE , DOUBLE PRECISION , REAL
Synonymous with FLOAT.

Floating Point Examples in Table Columns

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


desc table test_float;
```

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

Note
DOUBLE, FLOAT, DOUBLE PRECISION, and REAL columns are displayed as FLOAT, but stored as DOUBLE. This is a known issue in Snowflake.

Numeric Constants

Constants (also known as *literals*) refers to fixed data values. The following formats are supported for numeric constants:

```
[+-][digits][.digits][e[+-]digits]
```

Where:

- `+` or `-` indicates a positive or negative value. The default is positive.
- `digits` is one or more digits from 0 to 9.
- `e` (or `E`) indicates an exponent in scientific notation. At least one digit must follow the exponent marker if present.

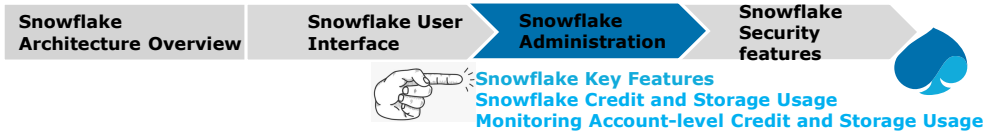
Following are all examples of supported numeric constants:

```
15
+1.34
0.2
15e-03
1.234E2
1.234E+2
-1
```


Agenda

- **Day 1: Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration**
  - ✓ **Snowflake Security features**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Day 1: Get started with Snowflake



- ✓ **Snowflake Administration**
  - Snowflake Key Features
  - Snowflake Credit and Storage Usage
  - Monitoring Account-level Credit and Storage Usage

This topic lists the notable/significant features supported in the current release. Note that it does not list every feature provided by Snowflake.

Security and Data Protection

•Choose the level of security you require for your Snowflake account, based on your **Snowflake Edition**.
•Choose the geographical location where your data is stored, based on your **Snowflake Region**.
•User authentication through standard user/password credentials.
•Enhanced authentication:
- Multi-factor authentication (MFA).
- Federated authentication and single sign-on (SSO) — requires Snowflake Enterprise Edition.

•Deployment inside a cloud platform VPC.
•Isolation of data via Amazon S3 policy controls.
•Automatic data encryption by Snowflake using Snowflake-managed keys.
•Object-level access control.

Standard and Extended SQL Support

•Most DDL and DML defined in SQL:1999, including:
- Database and schema DDL.
- Table and view DDL.
- Standard DML such as UPDATE, DELETE, and INSERT.
- DML for bulk data loading/unloading.
- Core data types.
- SET operations.
- CAST functions.

•Advanced DML such as multi-table INSERT, MERGE, and multi-merge.
•Transactions.
•Temporary and transient tables for transitory data.
•Statistical aggregate functions.
•Analytical aggregates (Group by cube, rollup, and grouping sets).
•Parts of the SQL:2003 analytic extensions:
- Windowing functions.
- Grouping sets.

•Scalar and tabular user-defined functions (UDFs), with support for both SQL and JavaScript.
•Information Schema for querying object and account metadata, as well as query and warehouse usage history data.

Tools and Interfaces

•Web-based GUI for account and general management, monitoring of resources and system usage, and querying data.
•SnowSQL (Python-based command line client).
•Virtual warehouse management from the GUI or command line, including creating, resizing (with zero downtime), suspending, and dropping warehouses.

Data Import and Export

•Support for bulk loading and unloading data into/out of tables, including:
- Load any data that uses a supported character encoding.
- Load data from compressed files.
- Load most flat, delimited data files (CSV, TSV, etc.).
- Load data files in JSON, Avro, ORC, Parquet, and XML format.
- Load from S3 data sources and local files using Snowflake web interface or command line client.

•Support for continuous bulk loading data from files:
- Use Snowpipe to load data in micro-batches from internal stages (i.e. within Snowflake) or external stages (i.e. in S3 or Azure).

Connectivity

•Broad ecosystem of supported 3rd-party partners and technologies.
•Support for using free trials to connect to selected partners.
•Extensive set of client connectors and drivers provided by Snowflake:
- Python connector
- Spark connector
- Node.js driver
- Go Snowflake driver
- .NET driver
- JDBC client driver
- ODBC client driver

Data Sharing A

•Support for sharing data with other Snowflake accounts: Provide data to other accounts to consume.
•Consume data provided by other accounts.

Snowflake Credit and Storage Usage

Viewing Credit Usage for Your Account

Users with the ACCOUNTADMIN role can use the Snowflake web interface or SQL to view monthly and daily credit usage for all the warehouses in your account.
To view warehouse credit usage for your account:

Web Interface:	Click on **Account** » **Billing & Usage**
SQL:	Query the WAREHOUSE_METERING_HISTORY table function (in the Information Schema).

Snowflake Server & Pricing (SaaS Model)– as on Sep 2018

Server Configuration

Warehouse Size	Servers / Cluster	Credits / Hour	Credits / Second	Additional Details
X-Small	1	1	0.0003	Default size for warehouses created using CREATE WAREHOUSE.
Small	2	2	0.0006	
Medium	4	4	0.0012	
Large	8	8	0.0024	
X-Large	16	16	0.0048	Recommended size (and default for warehouses created in the web interface).
2X-Large	32	32	0.0096	
3X-Large	64	64	0.0192	
4X-Large	128	128	0.0384	

Platform	Snowflake
Version	2.59.0
Instance Class	X-Large
Nodes	16
Cluster vCPUs	Unknown
Cluster RAM	Unknown
Storage	S3
Indicative Computing Cost	**$48 per hour ($3.00 per node)**

0-60 seconds:	1.07 credits
61 seconds:	1.08 credits
2 minutes:	2.15 credits
30 minutes:	32 credits
1 hour:	64 credits

Available Regions	Platform
US EAST	AWS
US WEST	AWS
Dublin	AWS
Frankfurt	AWS
Sydney	AWS
EASTUS2	AZURE

Pricing

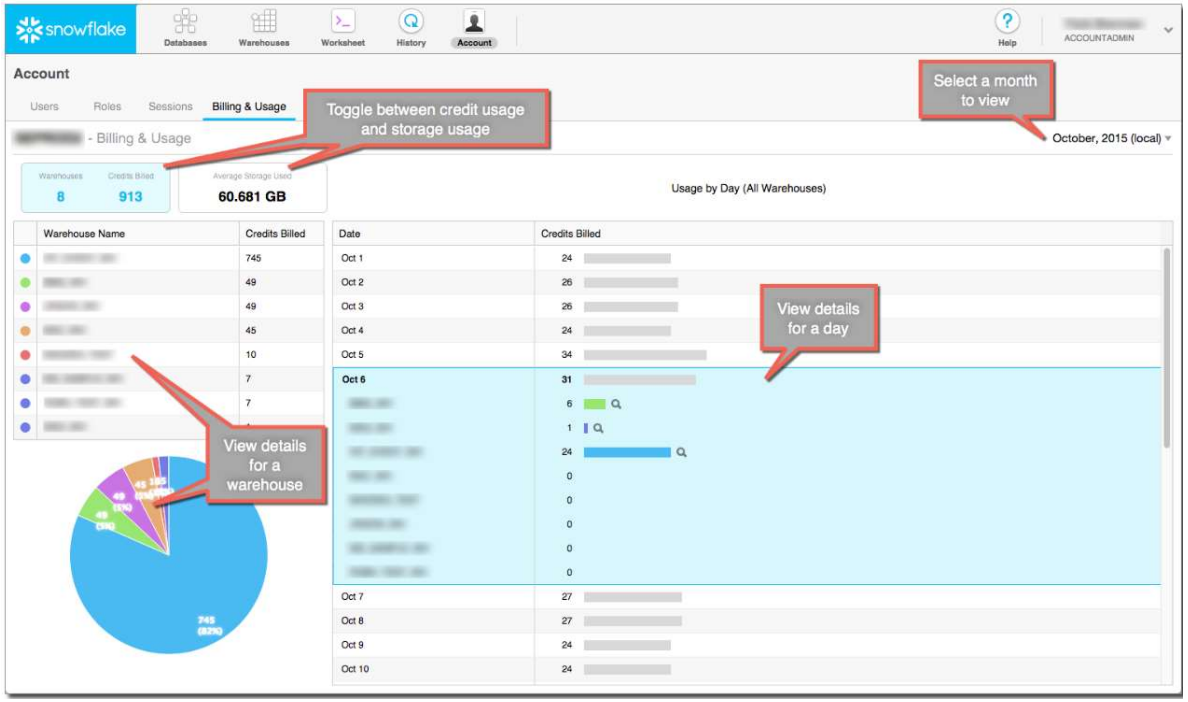
Snowflake On Demand	STANDARD	PREMIER	ENTERPRISE	ENTERPRISE FOR SENSITIVE DATA	VIRTUAL PRIVATE SNOWFLAKE
On Demand Storage per TB / mo	$40.00	$40.00	$40.00	$40.00	On contact
Compute cost per credit / per Hr	$2.00	$2.25	$3.00	$4.00	On contact
Snowflake Capacity					
Capacity Storage per TB / mo	$23.00	$23.00	$23.00	$23.00	On contact

* Cost may vary for regions

Monitoring Account-level Credit and Storage Usage

Billing & Usage tab available from the **Account** page in the Snowflake web interface

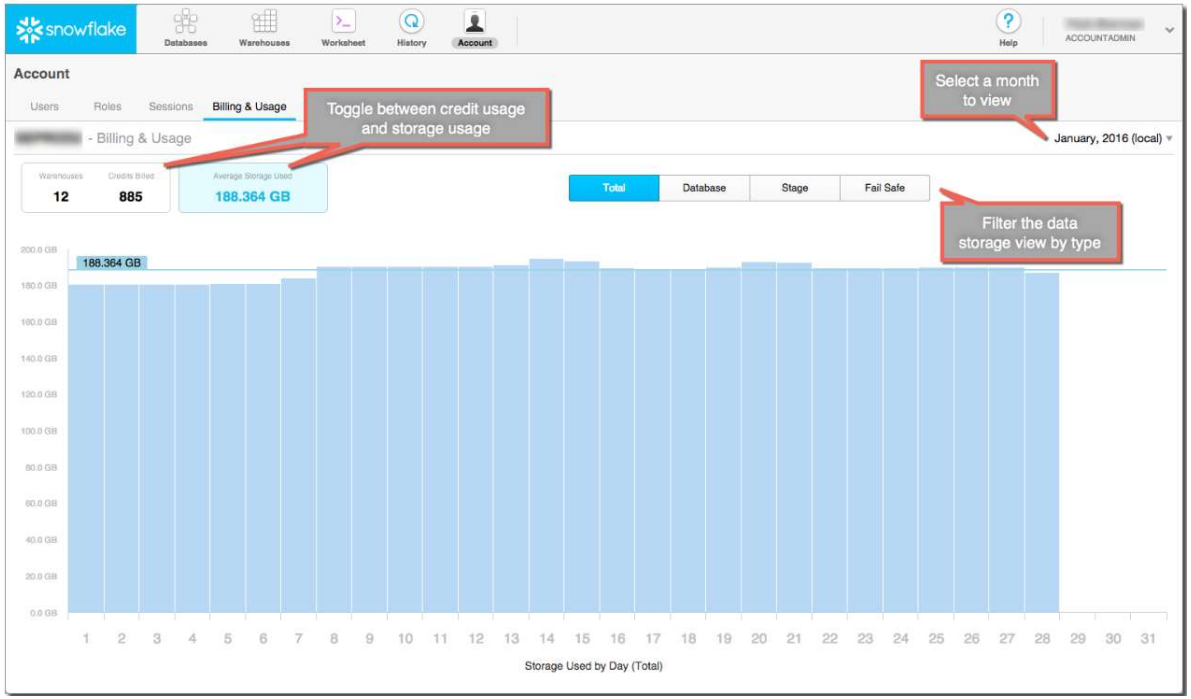
Viewing Credit Usage for Warehouses



In this view, you can:

- Select the month for which you wish to view credit usage.
- Click the name of a warehouse (or corresponding pie slice) to view daily credit usage for the selected warehouse.
- With no warehouse selected, click a day of the month to view a breakdown of credit usage by warehouse for the selected day.
- Use the **Download** button to output the information displayed on this page to a tab-delimited text file.

Viewing Data Storage Usage



In this view, you can:

- Select the month for which you wish to view storage usage.
- Click the filters on the top to display the storage usage by type of storage.
- Use the **Download** button to output the information displayed on this page to a tab-delimited text file. The storage units (MB, GB, TB, etc.) change dynamically depending on the amount of data stored.

Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

DATA WAREHOUSE SECURITY BY DESIGN

Industrial-strength Data Warehouse Security.
Deployed Securely in the Cloud.
Certified and Validated.
VIRTUAL PRIVATE SNOWFLAKE.
Managing Security in Snowflake

Snowflake follows best-in-class, standards-based practices to ensure your data and data warehouse security. Security was baked into Snowflake at the very beginning, not as an add-on or afterthought, removing the complexity and burden of enabling security from our customers.

Industrial-strength Data Warehouse Security

Snowflake provides the protection required of an enterprise-class data warehouse.

- Fine-grained, role-based access control for data and actions
- Always-on encryption of data stored in Snowflake
- Automatic protection against accidental or malicious loss of data
- AWS PrivateLink connectivity to utilize Snowflake without going over the public Internet.

Encryption of data

- An encrypted database uses an algorithm to convert plain text data into unusable cypher text.
- Database decryption keys are then deployed to convert the cypher text back to its original state.

Database Encryption at Snowflake:

Snowflake provides robust, enterprise-class data warehouse security that features round-the-clock data encryption via CloudHSM, automated data protection, granular role-based permissions, and AWS PrivateLink to access Snowflake outside the public Internet. This full set of data security features is not available from Hadoop and many other Big Data storage and warehousing platforms.

Industrial-strength Data Warehouse Security

Snowflake provides the protection required of an enterprise-class data warehouse.

- Fine-grained, **role-based access control** for data and actions
- Always-on **encryption of data** stored in Snowflake
- Automatic **protection** against accidental or malicious loss of data
- AWS **PrivateLink** connectivity to utilize Snowflake without going over the public Internet.

Data Protection

- Data protection is the process of safeguarding important information from corruption and/or loss.
- Data protection is one of the most important concerns of the modern data driven organization. As the amount of data being collected increases, the value of that information rises and its vulnerability to attack becomes more acute. The rise of cloud services hasn't fundamentally altered the sensitivity of data or information, but it has brought to the forefront the importance of data security.
- Snowflake is designed to protect user data against attacks on all levels of the data architecture, including the cloud platform. To this end, Snowflake implements two-factor authentication, (client-side) encrypted data import and export, secure data transfer and storage, and role-based access control for database objects. At all times, data is encrypted before being sent over the network, and before being written to local disk or shared storage (S3).

PRIVATELINK FOR SNOWFLAKE: NO INTERNET REQUIRED

- AWS recently announced PrivateLink, the newest generation of VPC Endpoints that allows direct and secure connectivity between AWS VPCs, without traversing the public Internet. We've been working closely with the AWS product team to integrate PrivateLink with Snowflake and we're excited to be among the first launch partners. By integrating with PrivateLink, we allow customers with strict security policies to connect to Snowflake without exposing their data to the Internet. We'll highlight how PrivateLink enhances our existing security capabilities, and how customers can easily set up PrivateLink with Snowflake.

Deployed Securely in the Cloud

- Snowflake customers can choose to run their cloud-built data warehouse on Amazon Web Services' or Microsoft Azure's highly secure infrastructure platforms.
- The Snowflake service runs inside a Virtual Private Cloud, with individual hosts protected by firewalls configured with the most stringent rules. All communication with the Snowflake service is protected at the network level using industrial-strength, secure protocols.

Certified and Validated

- Snowflake leverages established best practices for security controls as part of our security program. Snowflake works with **AICPA** (American Institute of Certified Public Accountants)-certified, third-party auditors to maintain security compliance and attestations including **SOC** 2 (system of service organization controls), Type II.
- Snowflake is also **PCI DSS** (The Payment Card Industry Data Security Standard) certified, FedRamp Ready, and **HIPAA** compliant, with the processes and controls in place required by the U.S. Health Insurance Portability and Accountability Act (HIPAA).

VIRTUAL PRIVATE SNOWFLAKE

The Most Secure Data Warehouse for Sensitive Data

- Enabling a secure data warehouse for your organization is straightforward with Snowflake. Create your own instance of Snowflake today. Start loading data and immediately see the benefits of the only data warehouse built for the cloud.

Managing Security in Snowflake

Industrial-strength Data Warehouse Security.
Deployed Securely in the Cloud.
Certified and Validated.
VIRTUAL PRIVATE SNOWFLAKE.
Managing Security in Snowflake

Administrative concepts and tasks associated with managing account, user, and data security in Snowflake. (i.e. users with the ACCOUNTADMIN, SYSADMIN, or SECURITYADMIN roles).

Summary of Security Features
List of security features, grouped by category (access, authentication, etc.).
•Network Policies
Concepts and tasks for controlling site access through network policies and IP whitelisting/blacklisting.
•AWS PrivateLink & Snowflake
Using AWS PrivateLink to secure private/direct communication between Snowflake and your other VPCs.
•Multi-Factor Authentication (MFA)
Concepts and tasks for managing and using multi-factor authentication for more secure user login.
•Federated Authentication & SSO
•Overview of Federated Authentication and SSO
•Access Control in Snowflake
•Overview of Access Control
•Access Control Privileges
•Configuring Snowflake to Use Federated Authentication
•Configuring Access Control
•Access Control Considerations
•Data Encryption
Concepts related to how Snowflake automatically encrypts your data, as well as additional features you can enable for enhanced levels of encryption.

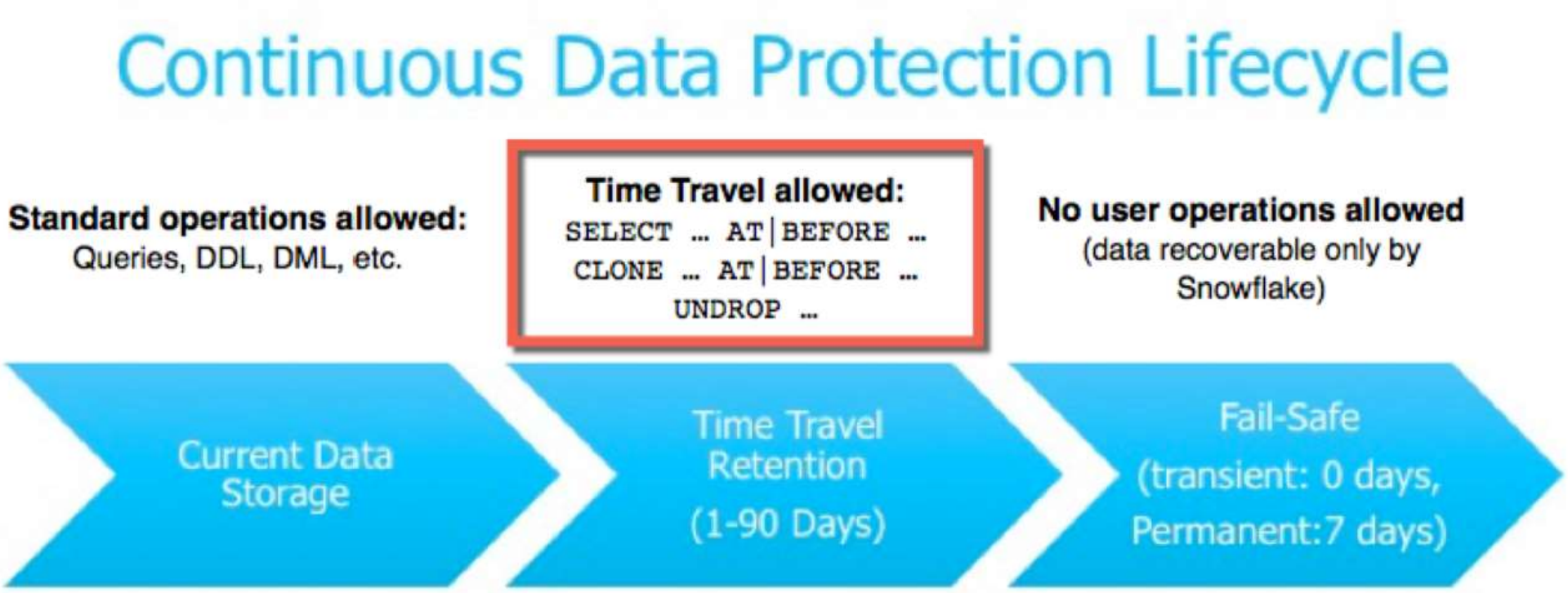
Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Share Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Data Backup & Recovery

- **Data Protection Lifecycle**
- Time Travel
- Failsafe
- Clone
- Recovery
- Continuous Data Protection

Data Protection Lifecycle



Time Travel - Data Retention Period

- DATA_RETENTION_TIME_IN_DAYS
- The standard retention period - 1 day (24 hours)
- Enterprise Edition
  - Temp/Transient: 0 or 1
  - Permanent : 0-90 days
- Disable and enable
- Override on Object level

```
create table mytable data_retention_time_in_days=90
alter table mytable set data_retention_time_in_days=30;
```


Time Travel – Query Historical Data

SELECT AT | BEFORE

- TIMESTAMP

```
select * from my_table at(timestamp => 'Mon, 01 May 2015 16:20:00 -0700'::timestamp);
```

- OFFSET

```
select * from my_table at(offset => -60*5);
```

- Query ID

```
select * from my_table before(statement => '8e5d0ca9-005e-44e6-b858-a8f5b37c5726');
```


Time Travel – Drop and Restore

- DROP table/schema/database
- SHOW [TABLES | SCHEMAS | DATABASES] HISTORY

Show tables history ; ➔ Show all available versions within retention

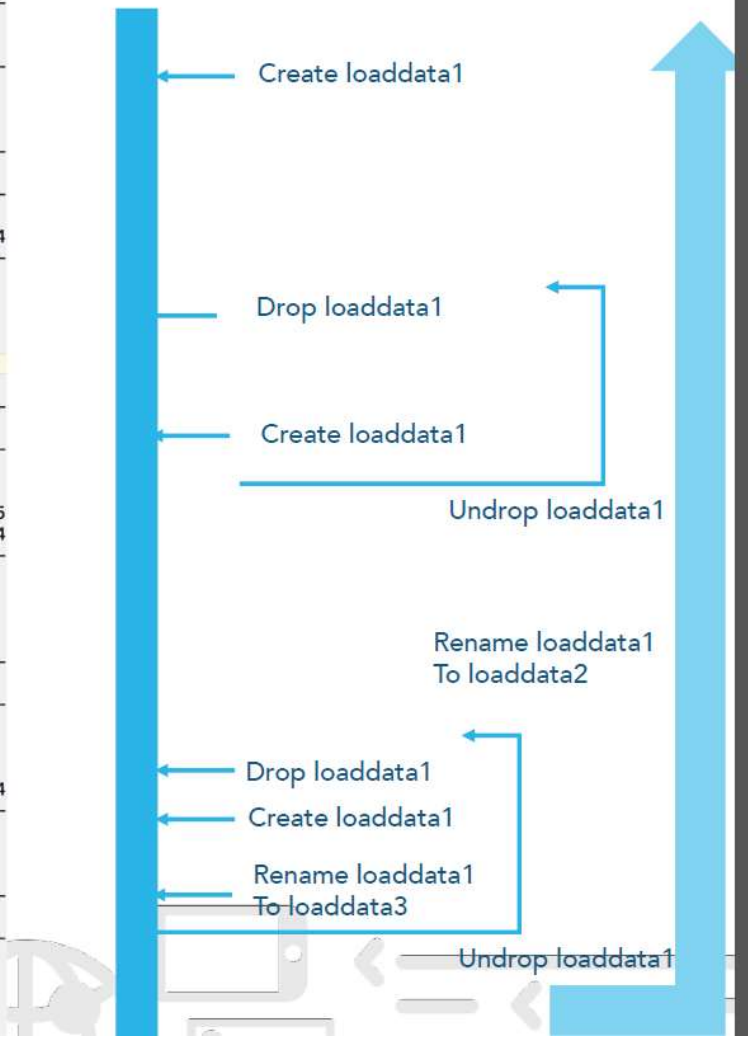
- UNDROP TABLE | SCHEMA | DATABASES to restore

```
undrop table mytable;
```

in the current schema or current database

- OWNERSHIP to UNDROP


```
show tables history;
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+
| created_on                    | name       | database_name | schema_name | kind  | retention_time | dropped_on                    |
|-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------|
| Tue, 17 Mar 2016 17:41:55 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:51:30 -0700 | PRODDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+

drop table loaddata1;
show tables history;
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+
| created_on                    | name       | database_name | schema_name | kind  | retention_time | dropped_on                    |
|-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------|
| Tue, 17 Mar 2016 17:51:30 -0700 | PRODDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:41:55 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | Fri, 13 May 2016 19:04        |
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+

create table loaddata1 (c1 number);
insert into loaddata1 values (1111), (2222), (3333), (4444);
drop table loaddata1;
create table loaddata1 (c1 varchar);
show tables history;
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+
| created_on                    | name       | database_name | schema_name | kind  | retention_time | dropped_on                    |
|-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------|
| Fri, 13 May 2016 19:06:01 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:51:30 -0700 | PRODDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Fri, 13 May 2016 19:05:32 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | Fri, 13 May 2016 19:05        |
| Tue, 17 Mar 2016 17:41:55 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | Fri, 13 May 2016 19:04        |
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+

alter table loaddata1 rename to loaddata3;
undrop table loaddata1;
show tables history;
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+
| created_on                    | name       | database_name | schema_name | kind  | retention_time | dropped_on                    |
|-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------|
| Fri, 13 May 2016 19:05:32 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Fri, 13 May 2016 19:06:01 -0700 | LOADDATA3  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:51:30 -0700 | PRODDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:41:55 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | Fri, 13 May 2016 19:04        |
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+

alter table loaddata1 rename to loaddata2;
undrop table loaddata1;
+-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------+
| created_on                    | name       | database_name | schema_name | kind  | retention_time | dropped_on                    |
|-------------------------------+------------+---------------+-------------+-------+----------------+-------------------------------|
| Tue, 17 Mar 2016 17:41:55 -0700 | LOADDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Fri, 13 May 2016 19:05:32 -0700 | LOADDATA2  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Fri, 13 May 2016 19:06:01 -0700 | LOADDATA3  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
| Tue, 17 Mar 2016 17:51:30 -0700 | PRODDATA1  | MYTESTDB      | PUBLIC      | TABLE | 1              | [NULL]                        |
```

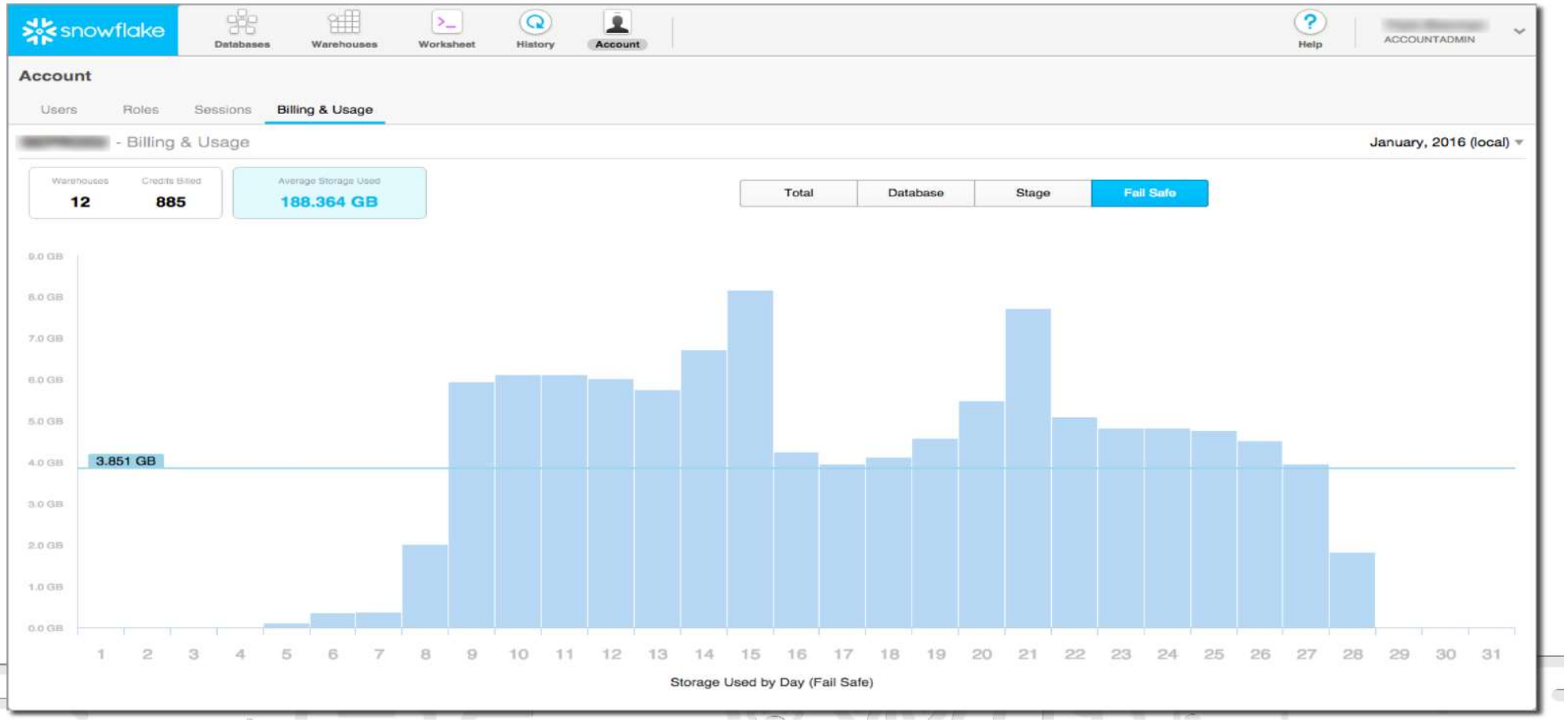


Failsafe – for Disaster recovery

- 7 Days
- Non-configurable
- No time travel
- Permanent objects

Continuous Data Protection Lifecycle

Failsafe – Storage



Storage Costs for Time Travel and Failsafe

- Calculated By 24 hour period unit
- the individual table rows for DML (Delta)
- Full table for DROP/TRUNCATE

Table Type	Time Travel Retention Period (Days)	Fail-safe Period (Days)	Min , Max Historical Data Maintained (Days)
Permanent	0 or 1 (for Snowflake Standard Edition)	7	**7 , 8**
	0 to 90 (for Snowflake Enterprise Edition)	7	**7 , 97**
Transient	0 or 1	0	**0 , 1**
Temporary	0 or 1	0	**0 , 1**

Clone

Databases, Schemas, Tables

```
CREATE [ OR REPLACE ] { DATABASE | SCHEMA | TABLE } [ IF NOT EXISTS ] <object_name>
  CLONE <source_object_name>
      [ { AT | BEFORE } ( { TIMESTAMP => <timestamp> | OFFSET => <time_difference> | STATEMENT => <id> } ) ]
  ...
```

Other Schema Objects

```
CREATE [ OR REPLACE ] { STAGE | FILE FORMAT | SEQUENCE } [ IF NOT EXISTS ] <object_name>
  CLONE <source_object_name>
  ...
```

- OWNERSHIP privilege on the source object
- ***not*** contribute to the overall data storage unless modify
- ***not*** include the load history of the source table
- not lock the object being cloned
- Not transfer the privileges

Clone Data Lifecycle Use Case

- CLONE for dev/test
  - Full logical copy of the data, but no extra storage
  - Test/dev operations against clone have no effect on original data
  - Security
    - Standard SQL role-based access control limits dev/test access to clone and not production data
    - Secure Views can permit role- or user-based obfuscation / masking / projection
- Business Impact – better quality code
  - Dev and test teams are working on data at scale, see true app performance
  - Full range of values means fewer surprises when app encounters live data

Recovery

- Time Travel
  - CREATE … CLONE
  - SELECT …. AT | BEFORE
  - UNDROP
- Failsafe
- Outside of Snowflake, i.e. S3 bucket in a different region or Blob storage on Azure → load/unload
- Cross-region/Cloud load/replication (roadmap)

Continuous Data Protection

- Network policies
- User verification required(MFA, Federated)
- Access Control to all objects via security roles
- Automatic encryption of data
- Maintenance of historical data

Break

Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Virtual Warehouses

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

A virtual warehouse, often referred to simply as a “warehouse”, is a cluster of compute resources in Snowflake. A warehouse provides the required resources, such as CPU, memory, and temporary storage, to perform the following operations in a Snowflake session:

•Executing SQL SELECT statements that require compute resources (e.g. retrieving rows from tables and views).

•Performing DML operations, such as:

- Updating rows in tables (DELETE , INSERT , UPDATE).
- Loading data into tables (COPY INTO <table>).
- Unloading data from tables (COPY INTO <location>).

•Overview of Warehouses
•Warehouse Considerations
•Monitoring Warehouse Load

Virtual Warehouses

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

A warehouse is defined by

- Its size
- Other properties that can be set to help control and automate warehouse activity.
- Warehouses can be started and stopped at any time.
- They can also be resized at any time, even while running, to accommodate the need for more or less compute resources, based on the type of operations being performed by the warehouse.

•Warehouse Size
  - Impact on Credit Usage and Billing
  - Impact on Query Processing

•Auto-suspension and Auto-resumption
•It will take care of Query Processing and Concurrency
•Warehouse Usage in Sessions
  - Default Warehouse for Users
  - Default Warehouse for Client Utilities/Drivers/Connectors

Overview of Warehouses

- Size specifies the number of servers that comprise each cluster in a warehouse

Warehouse Size	Servers / Cluster	Credits / Hour	Credits / Second	Additional Details
X-Small	1	1	0.0003	Default size for warehouses created using CREATE WAREHOUSE.
Small	2	2	0.0006	
Medium	4	4	0.0012	
Large	8	8	0.0024	
X-Large	16	16	0.0048	Recommended size (and default for warehouses created in the web interface).
2X-Large	32	32	0.0096	
3X-Large	64	64	0.0192	

Impact on Query Processing

- The size of a warehouse can impact the amount of time required to execute queries submitted to the warehouse, particularly for larger, more complex queries.
- query performance scales straight with warehouse size
- You can resize warehouse resources in case if query is running slowly
- The Additional resources will not impact on those servers which are already running

How Credit Usage and Billing impacts

- Snowflake utilizes per-second billing (with a 60-second minimum each time the warehouse starts)
- warehouses are billed ***only*** for the credits they actually consume.
- **For example-if** a 3X-Large warehouse utilizes a cluster of 64 servers. The corresponding total number of credits billed depends on how long the warehouse runs continuously (credit totals rounded to the nearest 100th):

0-60 seconds:	1.07 credits
61 seconds:	1.08 credits
2 minutes:	2.15 credits
30 minutes:	32 credits
1 hour:	64 credits

Auto-suspension and Auto-resumption

- The Auto suspension and resumption activity can be performed
  - **If auto-suspend is enabled-** The warehouse is automatically suspended if the warehouse is inactive for the specified period of time.
  - **If auto-resume is enabled**- The warehouse is automatically resumed when any statement that requires a warehouse is submitted ***and*** the warehouse is the current warehouse for the session.
- The above properties can be used for better utilization of warehouse and as per requirement
- Auto suspend ensures that the warehouse will not run in case if it is not used
- Auto Resume ensures that the warehouse will start as soon as we need to start.

Warehouse Query Processing and Concurrency

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

- The number of queries can be concurrently process is **determined by the size and complexity** of each query
- As soon as queries are submitted the warehouse calculates and reserves the compute resources needed to process each query
- If the warehouse does not have enough remaining resources to process a query, the query is queued, pending resources that become available as other running queries complete.
- Snowflake supports to set object-level parameters to set threshold using following properties
  - MAX_CONCURRENCY_LEVEL
  - STATEMENT_QUEUED_TIMEOUT_IN_SECONDS
  - STATEMENT_TIMEOUT_IN_SECONDS

Warehouse Usage in Sessions

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

- Every warehouse is associated with the session.
- The query will not get executed if the session is not associated with warehouse
- **Default Warehouse for Users**
  - Snowflake supports specifying a default warehouse for each individual user for fast query processing
  - Default warehouse can assigned when user are created
- **Default Warehouse for Client Utilities/Drivers/Connectors**
  - Snowflake clients (SnowSQL, JDBC driver, ODBC driver, Python connector, etc.) can also have a default warehouse

Precedence for Warehouse Defaults

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

- Snowflake determines the default warehouse for the session, the movement user connects warehouse

- Order of precedence for assigning warehouse to users
  - Default warehouse for the user **– this can be overridden**
  - Default warehouse in the configuration file for the client utility (SnowSQL, JDBC driver, etc.) – **this can also be overridden**
  - Default warehouse specified on the client command line or through the driver/connector parameters passed to Snowflake.

Warehouse Considerations

The keys to using warehouses effectively and efficiently are:

- Experiment with different types of queries and different warehouse sizes to determine the combinations that best meet your specific query needs and workload.
- Don’t focus on warehouse size. Snowflake utilizes per-second billing, so you can run larger warehouses (Large, X-Large, 2X-Large, etc.) and simply suspend them when not in use.

Note

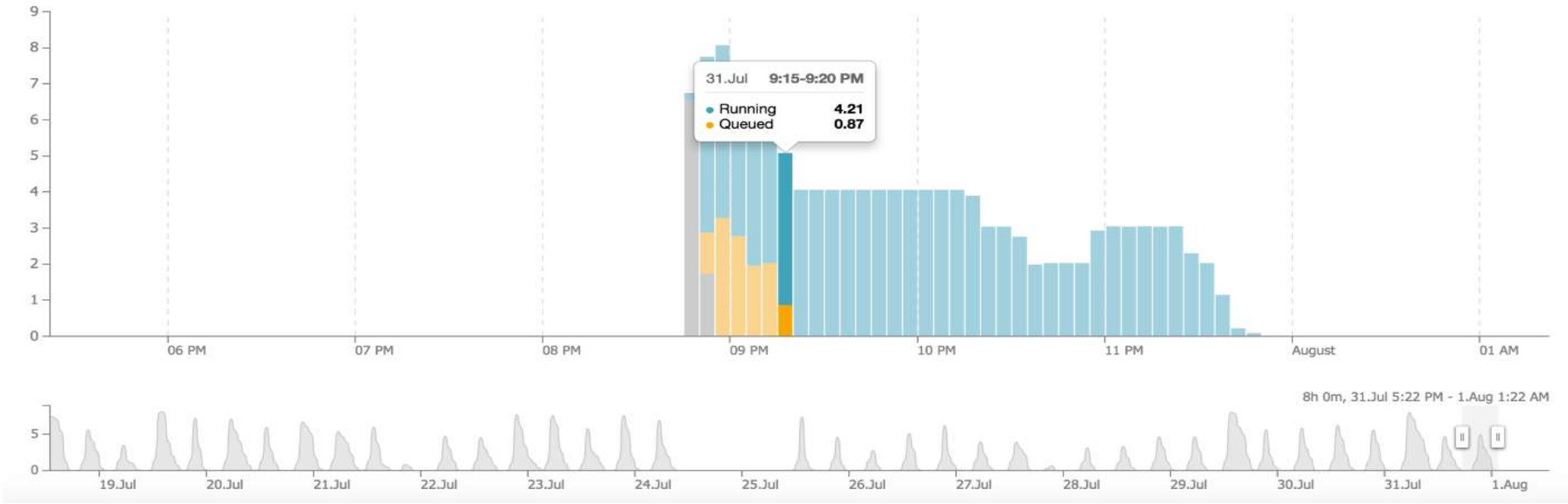
These guidelines and best practices apply to both single-cluster warehouses, which are standard for all accounts, and multi-cluster warehouses, which are available in Snowflake Enterprise Edition (and higher).

Monitoring Warehouse Load

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

The web interface provides a *query load* chart that depicts concurrent queries processed by a warehouse over a two-week period. Warehouse query load measures the average number of queries that were running or queued within a specific interval.

The **Warehouse Load Over Time** page appears with a bar chart and a slider for selecting the window of time to view in the chart. By default, the chai displays the past 8 hours in 5-minute intervals.



Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses -- Covered**
  - ✓ **Understanding Databases and Storage**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
- **Hands-On**

Databases, Tables & Views

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

- All data in Snowflake is maintained in databases.
- Each database consists of one or more schemas, which are logical groupings of database objects, such as tables and views.
- Snowflake does not place any hard limits on the number of databases, schemas (within a database), or objects (within a schema) you can create.

•Understanding Snowflake Table Structures
•Automatic Clustering
•Working with Temporary and Transient Tables
•Overview of Views
•Working with Secure Views
•Working with Materialized Views
•Data Storage Considerations A

Understanding Snowflake Table Structures

- All data in Snowflake is stored in database tables.
- logically structured as collections of columns and rows.
- ***Micro-partitions*** and ***data clustering***, two of the principal concepts utilized in Snowflake physical table structures.
- It help optimize table maintenance and query performance.

Micro-partitions

- Traditional data warehouses trust on static partitioning of large
- Snowflake has implemented a powerful and unique form of partitioning, called *micro-partitioning.*
- That delivers all the advantages of static partitioning without the known limitations, as well as providing additional significant benefits.

What are Micro-partitions?

- All data in Snowflake tables is automatically divided into micro-partitions, which are contiguous units of storage.
- Each micro-partition contains between 50 MB and 500 MB of uncompressed data.
- Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion.
- All DML operations (e.g. DELETE, UPDATE, MERGE) take advantage of the underlying micro-partition metadata to facilitate and simplify table maintenance. For example, some operations, such as deleting all rows from a table, are metadata-only operations.

Snowflake tracks metadata about all rows stored in a micro-partition, including:
•The range of values for each of the columns in the micro-partition.
•The number of distinct values.
•Additional properties used for both optimization and efficient query processing.

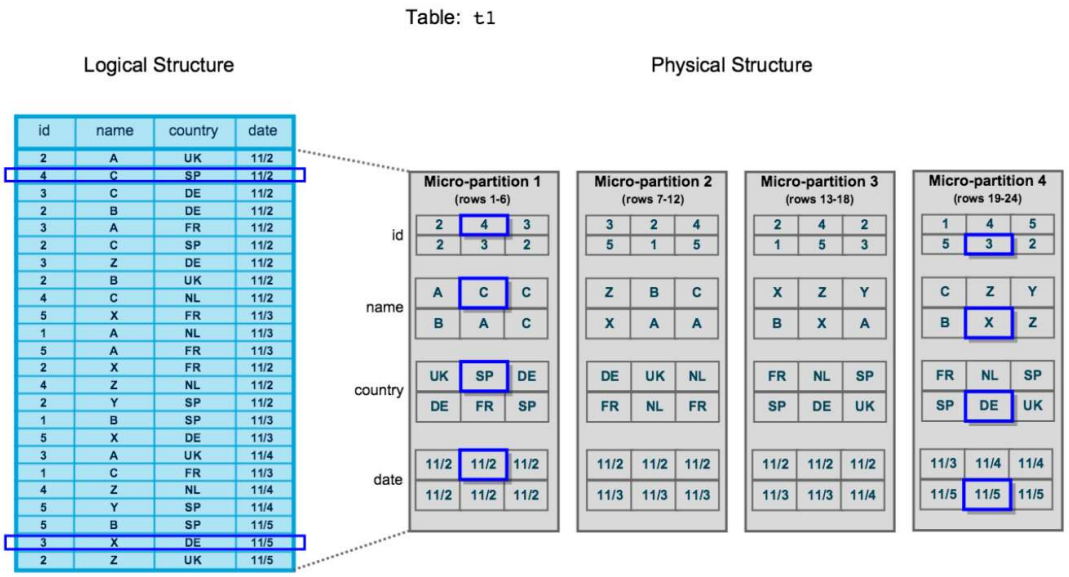
Note
Micro-partitioning is automatically performed on all Snowflake tables. Tables are transparently partitioned using the ordering that occurs when the data is inserted/loaded.

Understanding Snowflake Table Structures

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Data Clustering

Data is loaded into a Snowflake table, Snowflake co-locates column data with the same values in the same micro-partition, if possible.
Snowflake then leverages the metadata it transparently maintains for each table to avoid scanning micro-partitions during queries, significantly accelerating the performance of queries that reference these columns.
The following diagram illustrates a Snowflake table, `t1`, with 4 columns sorted by date:



id	name	country	date
2	A	UK	11/2
4	C	SP	11/2
3	C	DE	11/2
2	B	DE	11/2
3	A	FR	11/2
2	C	SP	11/2
3	Z	DE	11/2
2	B	UK	11/2
4	C	NL	11/2
5	X	FR	11/3
1	A	NL	11/3
5	A	FR	11/3
2	X	FR	11/2
4	Z	NL	11/2
2	Y	SP	11/2
1	B	SP	11/3
5	X	DE	11/3
3	A	UK	11/4
1	C	FR	11/3
4	Z	NL	11/4
5	Y	SP	11/4
5	B	SP	11/5
3	X	DE	11/5
2	Z	UK	11/5

The table consists of 24 rows stored across 4 micro-partitions, with the rows divided equally between each micro-partition. Within each micro-partition, the data is sorted and stored by column, which enables Snowflake to perform the following actions for queries on the table:

1. First cut back micro-partitions that are not needed for the query.
2. Then cut back by column within the remaining micro-partitions.

CREATE TABLE

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Creates a new table in the current/specified schema or replaces an existing table. A table can have multiple columns, with each column definition consisting of a name, data type, and optionally whether the column has:
•A default value and/or requires a value (NOT NULL).
•Any referential integrity constraints (primary key, foreign key, etc.).

In addition, this command supports the following variants:
•CREATE TABLE … AS SELECT (creates a populated table; also referred to as CTAS)
•CREATE TABLE … LIKE (creates an empty copy of an existing table)
•CREATE TABLE … CLONE (creates a clone of an existing table)

```
CREATE [ OR REPLACE ] [ { [ LOCAL | GLOBAL ] TEMP[ORARY] | VOLATILE } | TRANSIENT ] TABLE [ IF NOT EXISTS ] <table_name> ( <col_name> <col_type> [
{ DEFAULT <expr> | { AUTOINCREMENT | IDENTITY } [ ( <start_num> , <step_num> ) | START <num> INCREMENT <num> ] } ] /* AUTOINCREMENT / IDENTITY
supported only for numeric data types (NUMBER, INT, etc.) */ [ inlineConstraint ] [ , <col_name> <col_type> ... ] [ , outoflineConstraint ] [ ,
... ] ) [ CLUSTER BY ( <expr> [ , <expr> , ... ] ) ] [ STAGE_FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>' | TYPE = { CSV | JSON | AVRO |
ORC | PARQUET | XML } [ formatTypeOptions ] } ) ] [ STAGE_COPY_OPTIONS = ( copyOptions ) ] [ DATA_RETENTION_TIME_IN_DAYS = <num> ] [ COPY GRANTS ]
[ COMMENT = '<string_literal>' ]
```


Temporary and Transient Tables

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Temporary Tables:

- Snowflake supports creating temporary tables for storing non-permanent, transitory data (e.g. ETL data, session-specific data).
- Temporary tables only exist within the session and they are not visible to other users or sessions.
- Once the session ends, data stored in the table is purged completely from the system and, therefore, is not recoverable either by the user who created the table or Snowflake.

Creating a Temporary Table:

To create a temporary table, simply specify the TEMPORARY keyword (or TEMP abbreviation) in CREATE TABLE. For example:

```
create temporary table mytemptable (id number, creation_date date);
```

Transient Tables:

- Transient tables are similar to permanent tables.
- key difference that they do not have a Fail-safe period.
- As a result, transient tables are specifically designed for transitory data that needs to be maintained beyond each session.
- Does not need the same level of data protection and recovery provided by permanent tables.

```
create transient table mytranstable (id number, creation_date date);
```

Note
After creation, transient tables cannot be converted to any other table type.

Comparison of Table Types

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

The following table summarizes the differences between the three table types, particularly with regard to their impact on Time Travel and Fail-safe:

Type	Persistence	Time Travel Retention Period (Days)	Fail-safe Period (Days)
Temporary	Remainder of session	0 or 1 (default is 1)	0
Transient	Until explicitly dropped	0 or 1 (default is 1)	0
Permanent (Standard Edition)	Until explicitly dropped	0 or 1 (default is 1)	7
Permanent (Enterprise Edition and higher)	Until explicitly dropped	0 to 90 (default is configurable)	7

Overview of Views

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

What is a View?

A view allows the result of a query to be accessed as if it were a table. The query is specified in the CREATE VIEW statement.

For example:

```
create view doctor_view as select patient_name, diagnosis, treatment from hospital_table; create view accountant_view as
select patient_name, billing_address, cost from hospital_table;
```

A view can be used almost anywhere that a table can be used (e.g. in a join or a subquery):

For example:

```
select distinct diagnosis from doctor_view;
```

```
select treatment, cost from doctor_view as dv, accountant_view as av where av.patient_id = dv.patient_id;
```

Advantages of Views

Views help you to write clearer, more modular SQL code.

For example:

```
create view doctors as select * from employees where title = 'doctor'; create view nurses as select * from employees where
title = 'nurse'; create view medical_staff as select * from doctors union select * from nurses ;
```


Overview of Views

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

What is a View?

A view allows the result of a query to be accessed as if it were a table. The query is specified in the CREATE VIEW statement.

For example:

```
create view doctor_view as select patient_name, diagnosis, treatment from hospital_table; create view accountant_view as
select patient_name, billing_address, cost from hospital_table;
```

A view can be used almost anywhere that a table can be used (e.g. in a join or a subquery):

For example:

```
select distinct diagnosis from doctor_view;
```

```
select treatment, cost from doctor_view as dv, accountant_view as av where av.patient_id = dv.patient_id;
```

Advantages of Views

Views help you to write clearer, more modular SQL code.

For example:

```
create view doctors as select * from employees where title = 'doctor'; create view nurses as select * from employees where
title = 'nurse'; create view medical_staff as select * from doctors union select * from nurses ;
```


Secure Views & Materialized Views

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

- Views should be defined as secure when they are specifically designated for data privacy, i.e. to limit access to **sensitive data** that should not be exposed to all users of the underlying table(s).
- Secure views should ***not*** be used for views that are defined for query convenience
- **Bypasses** the **optimizations** used for regular views.
- This may result in some impact on query performance for secure views.

Example Secure View:

```
create or replace secure view myview comment='Test secure view' as select * from mytable;
```

Materialized Views:

- A materialized view is named as though it were a type of view, in many ways it behaves more like a table.
- A materialized view's results are stored, almost as though the results were a table.
- This allows faster access, but requires storage space.

Example Materialized View:

```
create or replace materialized view mv1 as select myresourceintensivefunction(binary_col) from table1; select * from mv1;
```

•Cloning of materialized views is not supported.
•Time Travel is not currently supported on materialized views.

Snowflake Understanding Storage Cost

Control Data Usage Cost:

- Carefully setup these features to influence cost. Understand that there are consequences of altering these features too, so special considerations are recommended before disabling or reducing them.
- **Persisted Query Results** can be Turned ON / OFF by setting the Parameter USE_CACHED_RESULT to True or False.
- **Time Travel** can be adjusted using Parameter DATA_RETENTION_TIME_IN_DAYS – Days can be specified between 0 to 90 based on Snowflake's Edition being used.

Tip: In scenarios where you don't need extensive history on Tables / Schemas / Databases: create them as **Transient** type to avoid their Time Travel beyond 1 day and any Fail safe copy. Also, consider a less Time Travel for DEV and Int Environments

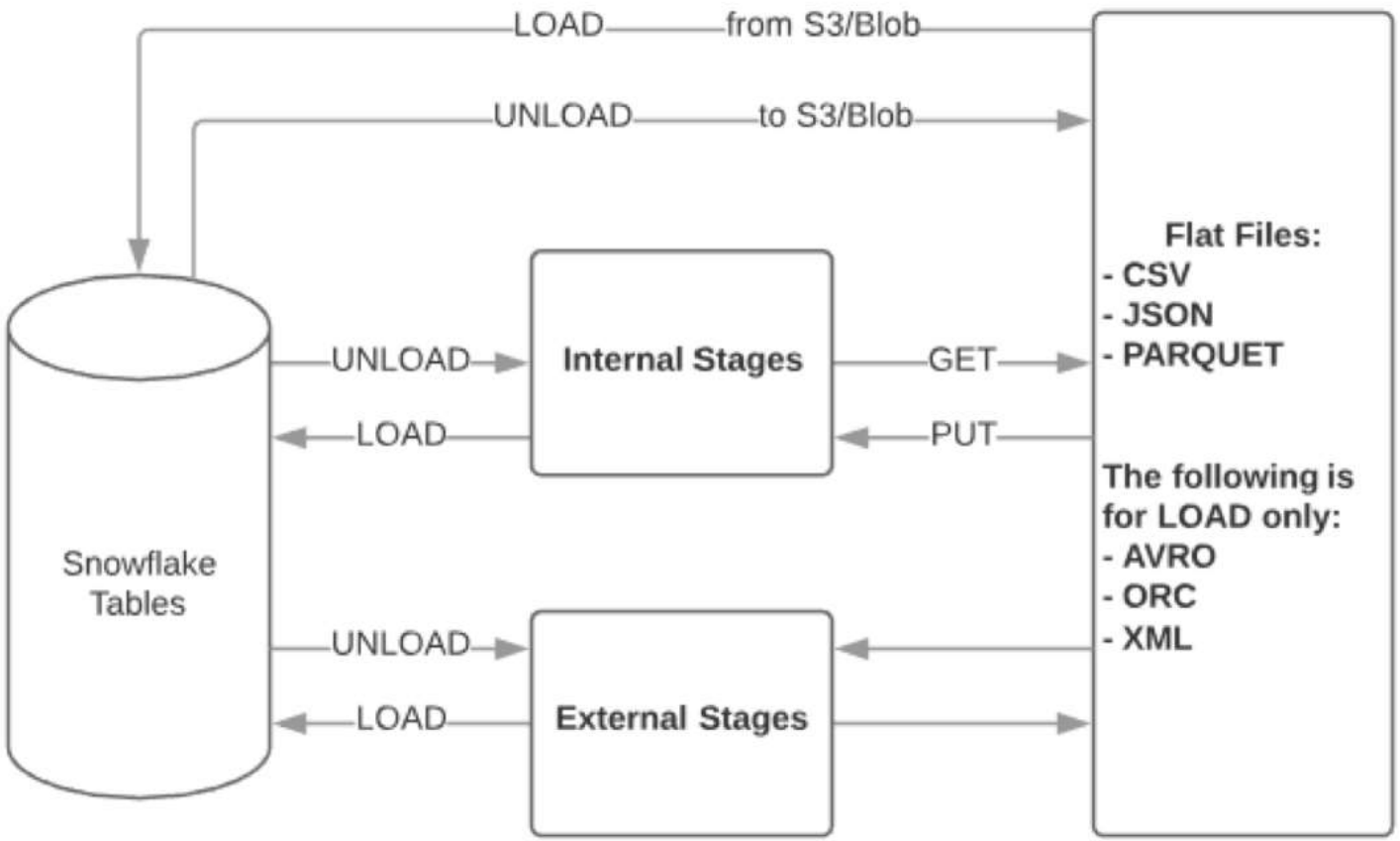
Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses -- Covered**
  - ✓ **Understanding Databases and Storage -- Covered**
  - ✓ **Load Data into Snowflake**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
- **Hands-On**

Loading Data in Snowflake

- Provides concepts and tasks for loading (i.e. importing) data into Snowflake database tables, as well as unloading (i.e. exporting) data from tables.
- Snowflake supports bulk import (i.e. loading) of data from one or more files into a table in Snowflake databases using the COPY command. Snowflake also supports loading limited amounts of data through the web interface.
- Snowflake supports loading data from the following file formats:
  - Any flat, delimited plain text format (CSV, TSV, etc.).
  - Semi-structured data in JSON, Avro, ORC, Parquet, or XML format (XML is currently supported as a preview feature).
- As data is loaded, Snowflake compresses the data and converts it into an optimized internal format for efficient storage, maintenance, and retrieval.

LOAD/UNLOAD FLOW



Overview of Data Loading

Bulk Loading Using COPY

- Bulk data loading into Snowflake tables using the COPY INTO <table> command.

Data loading is performed in two, separate steps:

1.Upload (i.e. stage) one or more data files into either an internal stage (i.e. within Snowflake) or an external location:

2.Use the COPY INTO <table> command to load the contents of the staged file(s) into a Snowflake database table. This step requires a running virtual warehouse that is also the current warehouse (i.e. in use) for the session. The warehouse provides the compute resources to perform the actual insertion of rows into the table.

Stage:

Internal:	Use the PUT command to stage the files.
External:	Currently, Amazon S3 and Microsoft Azure are the only services supported for staging external data. Snowflake assumes the files have already been staged in one of these locations. If they haven't been staged already, use the upload interfaces/utilities provided by the service that hosts the location.

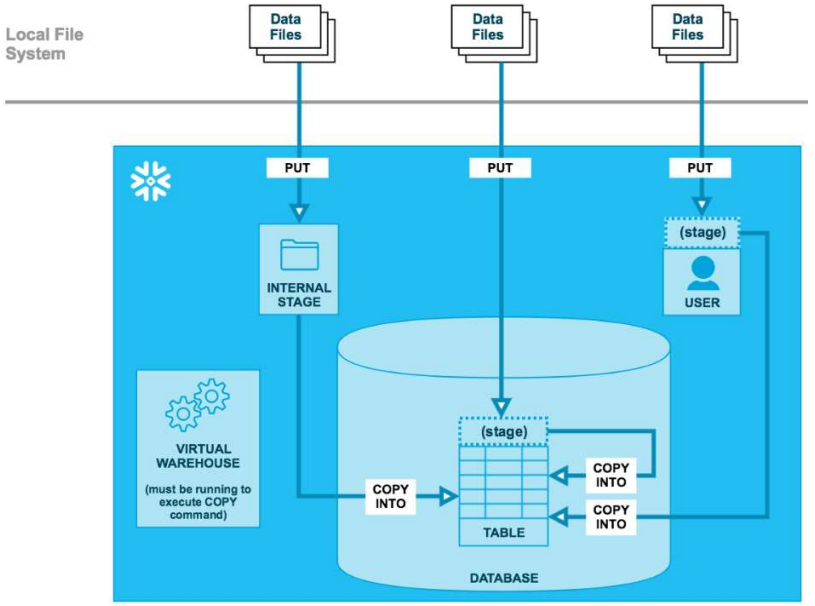
Tasks for Loading Data

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Bulk Loading from a Local File System Using COPY

Step 1: Upload (i.e. stage) one or more data files to a Snowflake stage (named internal stage or table/user stage) using the PUT command.

Step 2: Use the COPY INTO <table> command to load the contents of the staged file(s) into a Snowflake database table. Regardless of the stage you use, this step requires a running virtual warehouse that is also the current (i.e. in use) warehouse for the session. The warehouse provides the compute resources to perform the actual insertion of rows into the table.



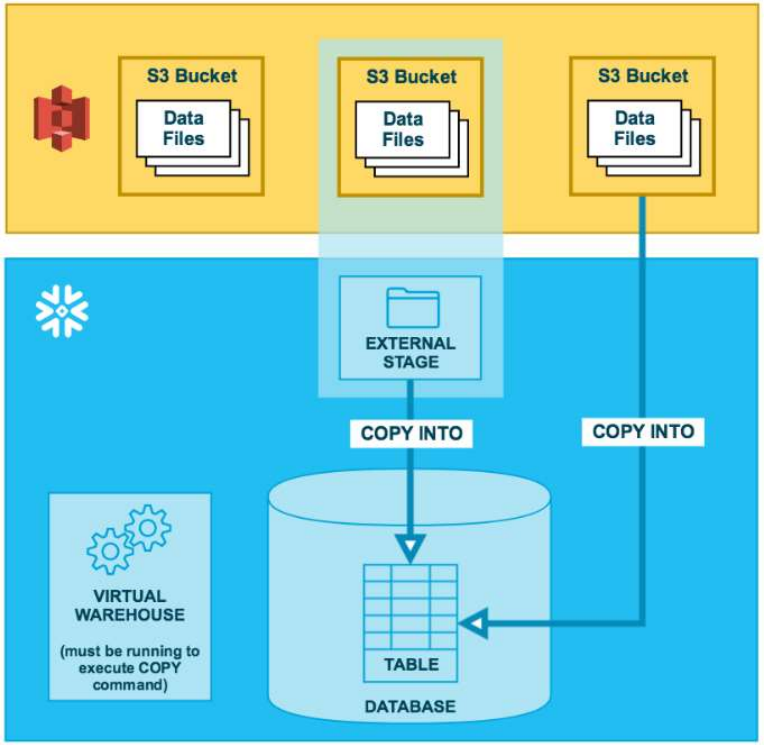
Tasks for Loading Data

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Bulk Loading from Amazon S3 Using COPY

Step 1: Snowflake assumes the data files have already been staged in an S3 bucket. If they haven't been staged yet, use the upload interfaces/utilities provided by AWS to stage the files.

Step 2: Use the COPY INTO <table> command to load the contents of the staged file(s) into a Snowflake database table. You can load directly from the bucket, but Snowflake recommends creating an external stage that references the bucket and using the external stage instead.
Regardless of the method you use, this step requires a running, current virtual warehouse for the session. The warehouse provides the compute resources to perform the actual insertion of rows into the table.



Tasks for Loading Data

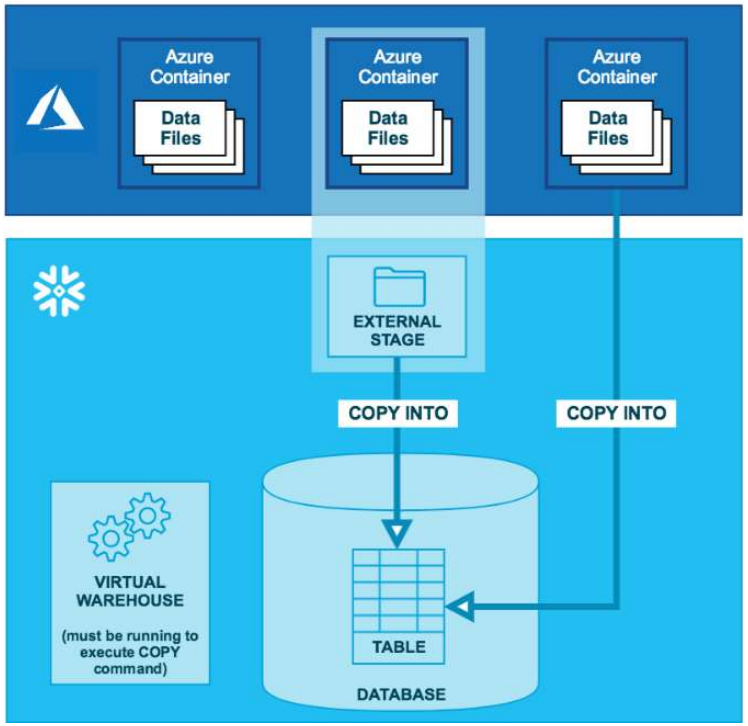
Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Bulk Loading from Microsoft Azure Using COPY

•Snowflake currently supports loading from/unloading to Blob storage only.

Step 1: Snowflake assumes the data files have already been staged in an Azure container. If they haven't been staged yet, use the upload interfaces/utilities provided by Microsoft to stage the files.

Step 2: Use the COPY INTO <table> command to load the contents of the staged file(s) into a Snowflake database table. You can load directly from the bucket, but Snowflake recommends creating an external stage that references the bucket and using the external stage instead.



Tasks for Loading Data

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Loading Continuously Using Snowpipe

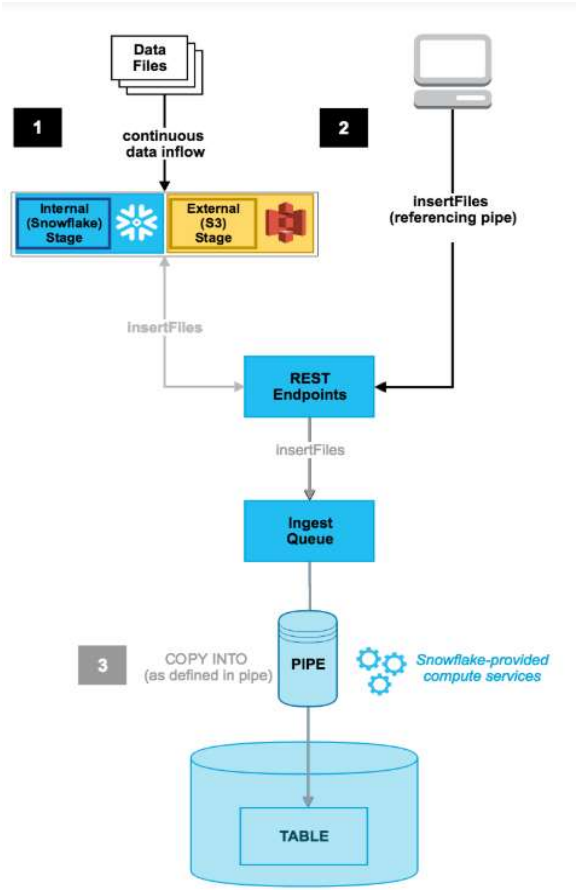
•Snowpipe is Snowflake's continuous data ingestion service.
•Snowpipe loads data within minutes after files are added to a stage and submitted for ingestion.
•The service can load data from any internal (i.e. Snowflake) or external (i.e. AWS S3 or Microsoft Azure) stage.
• In short, Snowpipe provides a "pipeline" for loading fresh data in micro-batches as soon as it's available.

How Does Snowpipe Work?

1.Data files are loaded in an internal (i.e. Snowflake) or external (i.e. S3 or Azure) stage.

2.A client calls the insertFiles endpoint with a list of files to ingest and a defined pipe. The endpoint moves these files to an ingest queue.

3.A Snowflake-provided virtual warehouse loads data from the queued files into the target table based on parameters defined in the specified pipe.



How Is Snowpipe Different from the COPY Command?

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

	COPY Command	Snowpipe
Authentication	Traditional username/password authentication.	***When calling the REST endpoints:*** Keypair-based authentication with JSON Web Token (JWT). JWTs are signed using a public/private key pair with RSA encryption.
Transactions	Adds data to a table in transactions alongside any other SQL statements submitted manually by users.	Adds data to a table in transactions controlled by Snowflake with no opportunity to involve other statements in the transaction.
Load History	Available upon completion of the COPY statement as statement results.	Must be requested from Snowflake via either a REST endpoint or a SQL table function.
Compute Resources	Requires a user-specified warehouse to execute COPY statements.	Uses Snowflake-supplied compute resources.
Cost	Billed for the amount of time each virtual warehouse is active.	Billed according to the compute resources used in the Snowpipe warehouse while loading the files.

Loading Using the Web Interface (Limited)

- Web interface provides a convenient wizard for loading limited amounts of data into a table from a small set of flat files.
- Behind the scenes, the wizard uses the PUT and COPY commands to load data;
- however, the wizard simplifies the data loading process by combining the two phases (staging files and loading data) into a single operation and deleting all staged files after the load completes.
- The wizard is only intended for loading small numbers of files of limited size (up to 50MB).

Step 1: Open the Load Data Wizard

1. Click on **Databases** » **Tables**.
2. Either:
   - Click on a table row to select it, then click the **Load Data** button.

   or

   - Click a table name to open the table details page, then click the **Load Table** button.

The **Load Data** wizard opens. The wizard will load data into the table you selected.

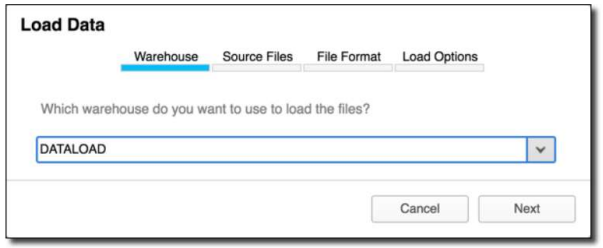
Loading Using the Web Interface (Limited)

Understanding Virtual Warehouses
Understanding Databases and Storage
Snowflake Data Loading
Loading Data Faster with Snowpipe
Suggested Documentation

Step 2: Select a Warehouse

1. Select a warehouse from the dropdown list.

   The list includes any warehouse on which you have the USAGE privilege. Snowflake will use this warehouse to load data into the table.



2. Click **Next**.

Step 3: Select Source Files

Important

Remember, the wizard is designed to load small amounts of data. Your individual data files should each be smaller than 50MB in size.

You can choose to load data from files on your local machine or files already staged in an existing Amazon S3 bucket.

- Loading from your local machine
- Loading from an S3 bucket
- Loading from an Azure container

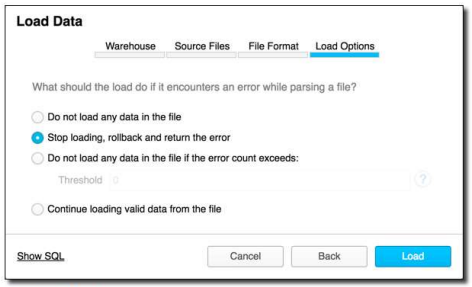
Step 4: Select a File Format

The dropdown list allows you to select a named set of options that describes the format of your data files.

- Selecting an existing named file format
- Creating a new named file format

Step 5: Select Load Options

1. Specify how Snowflake should behave if errors in the data files are encountered.



2. Click the **Load** button.

   Snowflake loads the data into your selected table using the warehouse you selected.

Note

If the warehouse is not currently running, resuming the warehouse could take some time (up to 5 minutes), in addition to the time required for loading.

3. Click the **OK** button.

Overview of Data Unloading

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

Bulk Unloading Process:

Step 1: Use the COPY INTO <location> command to copy the data from the Snowflake database table into one or more files in a Snowflake or external stage.

Step 2: Download the file from the stage:
- From a Snowflake stage, use the GET command to download the data file(s).
- From S3, use the interfaces/tools provided by Amazon S3 to get the data file(s).
- From Azure, use the interfaces/tools provided by Microsoft Azure to get the data file(s).

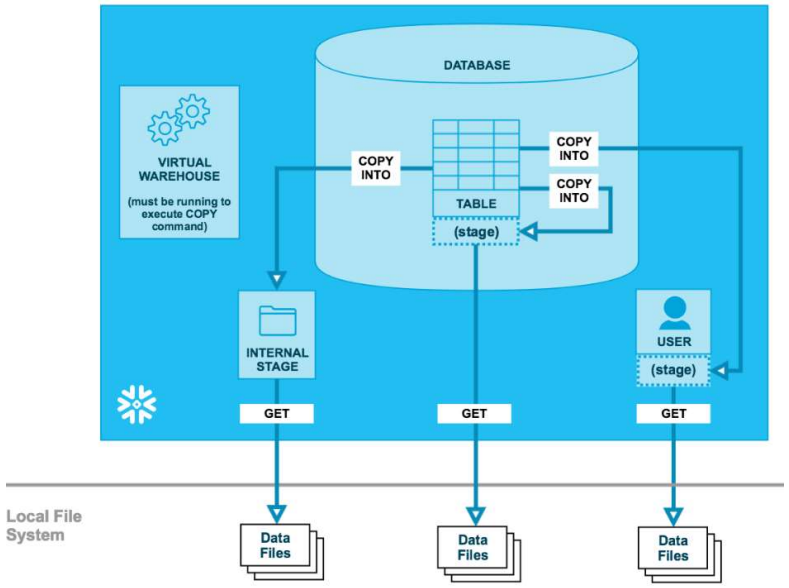
Bulk Unloading into Single or Multiple Files:

- The COPY INTO <location> command provides a copy option (SINGLE) for unloading data into a single file or multiple files.
- The default is SINGLE = FALSE (i.e. unload into multiple files).
- Snowflake assigns each file a unique name.
- Snowflake prefixes the generated filenames with data_. e.g. data_stats_0_1_0.

Unloading into a Snowflake Stage

Step 1: Use the COPY INTO <location> command to copy the data from the Snowflake database table into one or more files in a Snowflake stage. In you the command, you specify the stage (named stage or table/user stage) where the files are written.
Regardless of the stage you use, this step requires a running, current virtual warehouse for the session. The warehouse provides the compute resources to write rows from the table.

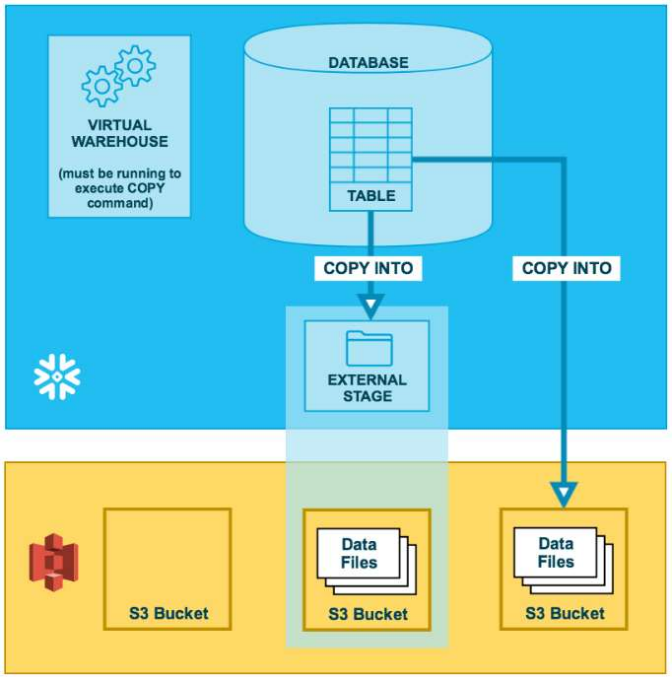
Step 2: Use the GET command to download the data files to your local file system.



Unloading into Amazon S3

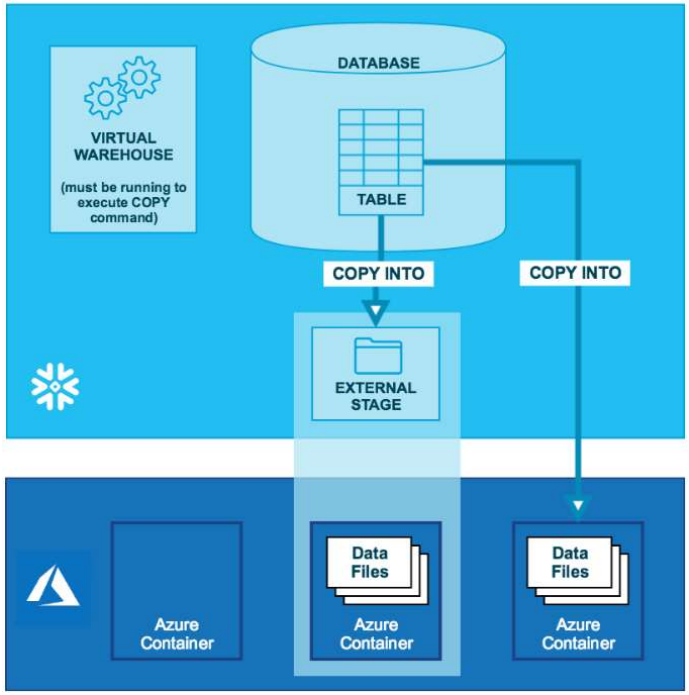
Step 1: Use the COPY INTO <location> command to copy the data from the Snowflake database table into one or more files in an S3 bucket. In the command, you specify a named external stage object that references the S3 bucket (recommended) or you can choose to unload directly to the bucket by specifying the URI and security credentials (if required) for the bucket.
Regardless of the method you use, this step requires a running, current virtual warehouse for the session. The warehouse provides the compute resources to write rows from the table.

Step 2: Use the interfaces/tools provided by Amazon to download the files from the S3 bucket.



Unloading into Amazon Azure

Step 1:	Use the COPY INTO <location> command to copy the data from the Snowflake database table into one or more files in an Azure container bucket. In the command, you specify a named external stage object that references the Azure container (recommended) or you can choose to unload directly to the container by specifying the URI and security credentials (if required) for the container. Regardless of the method you use, this step requires a running, current virtual warehouse for the session. The warehouse provides the compute resources to write rows from the table.
Step 2:	Use the interfaces/tools provided by Microsoft to download the files from the Azure container.



Summary of Data Unloading Features

Unloaded Data File Details

Feature	Supported	Notes
Location of files	Local files	Files are first unloaded to a Snowflake internal location, then can be downloaded locally using PUT.
	Files in AWS S3	Files can be unloaded directly to any user-supplied bucket in Amazon S3, then can be downloaded locally using AWS utilities.
	Files in Microsoft Azure	Files can be unloaded directly to any user-supplied container in Microsoft Azure, then can be downloaded locally using Azure utilities.
File formats	CSV and any other delimited formats (e.g. TSV)	Any single-character delimiter is supported; default is comma (i.e. CSV).
	JSON	
	Parquet	
File encoding	UTF-8	Unloaded files are encoded using UTF-8.

Summary of Data Unloading Features

Compression of Unloaded Data Files

Location of Files	Supported	Notes
Interal or external location	gzip	By default, all unloaded data files are compressed using gzip, unless compression is explicitly disabled or one of the other supported compression methods is explicitly specified.
	bzip2	
	Brotli	Note that support for Brotli and ZStandard compression of unloaded files is currently a preview feature.
	Zstandard	

Encryption of Unloaded Data Files

Location of Files	Supported	Notes
Internal location	128-bit or 256-bit keys	All data files unloaded to Snowflake internal locations are automatically encrypted using 128-bit keys. The files are unencrypted when they are downloaded to the local directory. 256-bit keys can be enabled (for stronger encryption); however, additional configuration is required.
External location	User-supplied key	Data files unloaded to S3 or Azure can be encrypted if a security key (for encrypting the files) is provided to Snowflake.

SQL Operator & Function Reference

Table Functions

Category/Sub-category	Functions
Table Functions	
Data Loading	VALIDATE
Data Generation	GENERATOR
Semi-structured Queries	FLATTEN
Query Results	RESULT_SCAN
Table Functions (Information Schema)	
Queries	QUERY_HISTORY , QUERY_HISTORY_BY_* (SESSION | USER | WAREHOUSE)
Warehouse & Storage Usage	DATABASE_STORAGE_USAGE_HISTORY , STAGE_STORAGE_USAGE_HISTORY , WAREHOUSE_METERING_HISTORY
Data Loading & Transfer	COPY_HISTORY , DATA_TRANSFER_HISTORY , PIPE_USAGE_HISTORY
User Login	LOGIN_HISTORY , LOGIN_HISTORY_BY_USER
SQL UDTFs (User-Defined Table Functions)	SQL or JavaScript

Operators

Category/Sub-category	Operators
Arithmetic Operators	+ , - , * , / , %
Comparison Operators	= , != , <> , < , <= , > , >=
Logical/Boolean Operators	AND , NOT , OR

User-defined Functions (UDFs)

In addition to the system-defined functions provided by Snowflake, users can create functions. Snowflake supports two types of UDFs:

Type	Notes
SQL	SQL UDFs can be defined to return either scalar or table output.
JavaScript	JavaScript UDFs can be defined to return either scalar or table output.

SQL Operator & Function Reference

Scalar Functions

Category/Sub-category	Functions
Bitwise Expression Functions	BITAND , BITNOT , BITOR , BITSHIFTLEFT , BITSHIFTRIGHT , BITXOR (see also: Bitwise Aggregation Functions)
Conditional Expression Functions	[NOT] BETWEEN , BOOLAND , BOOLNOT , BOOLOR , BOOLXOR , CASE , COALESCE , DECODE , EQUAL_NULL , GREATEST , IFF , IFNULL , [NOT] IN , IS [NOT] DISTINCT FROM , IS [NOT] NULL , IS_NULL_VALUE , LEAST , NULLIF , NVL , NVL2 , REGR_VALX , REGR_VALY , ZEROIFNULL
Context Functions	
General	CURRENT_* (CLIENT | DATE | TIME | TIMESTAMP | VERSION) , LOCALTIME , LOCALTIMESTAMP
Session	CURRENT_* (ROLE | SESSION | STATEMENT | TRANSACTION | USER) , LAST_QUERY_ID , LAST_TRANSACTION
Session Object	CURRENT_* (DATABASE | SCHEMA | SCHEMAS | WAREHOUSE)
Numeric Functions	
Rounding & Truncation	ABS , CEIL , FLOOR , MOD , ROUND , SIGN , TRUNC / TRUNCATE
Exponent & Root	CBRT , EXP , POW / POWER , SQRT , SQUARE
Logarithmic	LN , LOG
Trigonometric	ACOS , ACOSH , ASIN , ASINH , ATAN , ATAN2 , ATANH , COS , COSH , COT , DEGREES , HAVERSINE , PI , RADIANS , SIN , SINH , TAN , TANH
String & Binary Functions	
General	ASCII , BIT_LENGTH , CHARINDEX , CHR / CHR , CONCAT / || , CONTAINS , EDITDISTANCE , ENDSWITH , ILIKE , INITCAP , INSERT , LEFT , LENGTH , LIKE , LOWER , LPAD , LTRIM , OCTECT_LENGTH , PARSE_IP , PARSE_URL , POSITION , REPEAT , REPLACE , REVERSE , RIGHT , RPAD , RTRIM , RTRIMMED_LENGTH , SPACE , SPLIT , SPLIT_PART , STARTSWITH , SUBSTR / SUBSTRING , TRANSLATE , TRIM , UPPER, UUID_STRING
Encode & Decode	BASE64_DECODE_BINARY , BASE64_DECODE_STRING , BASE64_ENCODE , HEX_DECODE_BINARY , HEX_DECODE_STRING , HEX_ENCODE , TRY_* (decode binary and string functions)
Crytographic Hash	MD5 / MD5_HEX , MD5_BINARY , SHA1 / SHA1_HEX , SHA1_BINARY , SHA2 / SHA2_HEX , SHA2_BINARY

SQL Operator & Function Reference

Scalar Functions

String Functions (Regular Expressions)	REGEXP , REGEXP_COUNT , REGEXP_INSTR , REGEXP_LIKE , REGEXP_REPLACE , REGEXP_SUBSTR , RLIKE
Date & Time Functions	
Construction	DATE_FROM_PARTS / DATEFROMPARTS , TIME_FROM_PARTS / TIMEFROMPARTS , TIMESTAMP_FROM_PARTS / TIMESTAMPFROMPARTS
Extraction	DATE_PART , DAY , DAYNAME , DAYOFMONTH , DAYOFWEEK , DAYOFWEEKISO , DAYOFYEAR , EXTRACT , HOUR , LAST_DAY , MINUTE , MONTH , MONTHNAME , QUARTER , SECOND, WEEK , WEEKOFYEAR , WEEKISO , YEAR , YEAROFWEEK , YEAROFWEEK_ISO
Addition/Subtraction	ADD_MONTHS , DATEADD, DATEDIFF , TIMEADD, TIMEDIFF , TIMESTAMPADD , TIMESTAMPDIFF
Truncation	DATE_TRUNC , TRUNC
Type Conversion	TO_DATE , TO_TIME , TO_TIMESTAMP , TO_TIMESTAMP_* (LTZ | NTZ | TZ)
Time Zone	CONVERT_TIMEZONE
Semi-structured Data Functions	
Parsing	ARRAYS_OVERLAP , CHECK_JSON , CHECK_XML , PARSE_JSON , PARSE_XML , STRIP_NULL_VALUE
Array & Object	ARRAY_AGG , ARRAY_APPEND , ARRAY_CAT , ARRAY_COMPACT , ARRAY_CONSTRUCT , ARRAY_CONSTRUCT_COMPACT , ARRAY_CONTAINS , ARRAY_INSERT , ARRAY_POSITION , ARRAY_PREPEND , ARRAY_SIZE , ARRAY_SLICE , ARRAY_TO_STRING , OBJECT_AGG , OBJECT_CONSTRUCT , OBJECT_INSERT , OBJECT_DELETE
Data Extraction	FLATTEN , GET , GET_PATH , XMLGET
Casts	AS_* (all data types) , TO_ARRAY , TO_JSON , TO_OBJECT , TO_VARIANT, TO_XML
Type Predicates	IS_* (all data types) , TYPEOF
Conversion Functions	CAST , TO_* (all supported Snowflake data types) , TRY_CAST , TRY_TO_* (numeric, Boolean, date & time data types)
Aggregate Functions	
Utilities	GROUPING , GROUPING_ID
General	ANY_VALUE , AVG , CORR , COUNT , COVAR_POP , COVAR_SAMP , HASH_AGG , LISTAGG , MAX , MEDIAN , MIN , PERCENTILE_CONT , PERCENTILE_DISC , STDDEV / STDDEV_POP , STDDEV_SAMP , SUM , VAR_POP / VARIANCE_POP , VAR_SAMP / VARIANCE_SAMP / VARIANCE
Bitwise	BITAND_AGG , BITOR_AGG , BITXOR_AGG
Semi-structured Data	ARRAY_AGG , OBJECT_AGG
Linear Regression	REGR_AVGX , REGR_AVGY , REGR_COUNT , REGR_INTERCEPT , REGR_R2 , REGR_SLOPE , REGR_SXX , REGR_SXY , REGR_SYY
Cardinality Estimation	APPROX_COUNT_DISTINCT , HLL , HLL_ACCUMULATE , HLL_COMBINE , HLL_ESTIMATE , HLL_EXPORT , HLL_IMPORT
Similarity Estimation	APPROXIMATE_JACCARD_INDEX , APPROXIMATE_SIMILARITY , MINHASH , MINHASH_COMBINE
Frequency Estimation	APPROX_TOP_K , APPROX_TOP_K_ACCUMULATE , APPROX_TOP_K_COMBINE , APPROX_TOP_K_ESTIMATE
Percentile Estimation	APPROX_PERCENTILE , APPROX_PERCENTILE_ACCUMULATE , APPROX_PERCENTILE_COMBINE , APPROX_PERCENTILE_ESTIMATE
Analytic / Window Functions	CUME_DIST , DENSE_RANK , FIRST_VALUE , LAG , LAST_VALUE , LEAD , NTH_VALUE , NTILE , PERCENT_RANK , RANK , ROW_NUMBER , WIDTH_BUCKET
Miscellaneous Functions	
System	SYSTEM$ABORT_SESSION , SYSTEM$ABORT_TRANSACTION , SYSTEM$CANCEL_ALL_QUERIES , SYSTEM$CANCEL_QUERY , SYSTEM$CLUSTERING_DEPTH , SYSTEM$CLUSTERING_INFORMATION , SYSTEM$CLUSTERING_RATIO , SYSTEM$LAST_CHANGE_COMMIT_TIME , SYSTEM$PIPE_STATUS , SYSTEM$TYPEOF , SYSTEM$WAIT
Utility	GET_DDL , HASH
Data Generation	RANDOM , SEQ1 / SEQ2 / SEQ4 / SEQ8 , UUID_STRING
Random Distribution	NORMAL , RANDSTR , UNIFORM , ZIPF

Functions (by Category)

Provides links to the system-defined scalar functions, grouped by category:

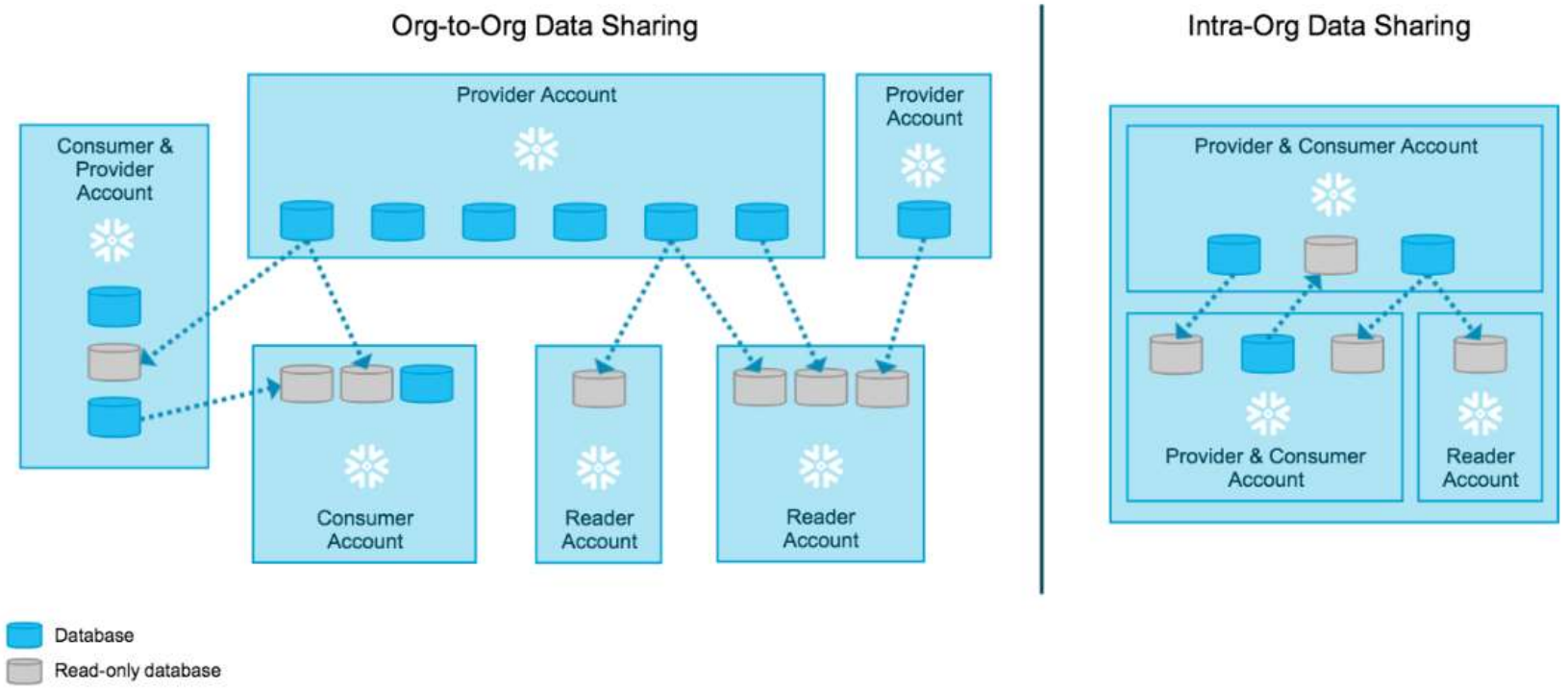
Category	Description
Bitwise Expression Functions	Perform bitwise operations on expressions.
Conditional Expression Functions	Manipulate conditional expressions.
Context Functions	Provide contextual information about the current environment, session, and object.
Numeric Functions	Perform rounding, truncation, exponent, root, logarithmic, and trigonometric operations on numeric values.
String & Binary Functions	Manipulate and transform string input.
String Functions (Regular Expressions)	Subset of strings functions for performing operations on items that match a regular expression.
Date & Time Functions	Manipulate dates, times, and timestamps.
Semi-structured Data Functions	Work with semi-structured data (JSON, Avro, etc.).
Conversion Functions	Convert expressions from one data type to another data type.
Aggregate Functions	Operate on values across rows to perform operations including counts, distincts, mathematical operations, approximate cardinality estimation, etc.
Analytic / Window Functions	Compute aggregates over a group (i.e. window) of rows.
Miscellaneous Functions	Utility functions, data generation functions, and random distribution functions.

Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses -- Covered**
  - ✓ **Understanding Databases and Storage -- Covered**
  - ✓ **Load Data into Snowflake -- Covered**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Query Data in Snowflake**
  - ✓ **Connecting to Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Introduction to Data Sharing

- Data Sharing enables account-to-account sharing of data through Snowflake database tables, secure views, and secure UDFs
- The principle participants in any Data Sharing relationship are the *provider* and one or more *consumers*. Snowflake enables the sharing of databases through *shares*, which are created by data providers and "imported" by data consumers.



Sample Data Sets

- Snowflake provides sample data sets, such as the industry-standard TPC-DS and TPC-H benchmarks, for evaluating and testing a broad range of Snowflake's SQL support.
- Sample data sets are provided in a database named SNOWFLAKE_SAMPLE_DATA that has been shared with your account from the Snowflake SFC_SAMPLES account.

Sample Data: TPC-DS

- TPC-DS models the decision support functions of a retail product supplier.
- Snowflake provides both 10TB and 100TB versions of TPC-DS, in schemas named TPCDS_SF10TCL and TPCDS_SF100TCL, respectively, within the SNOWFLAKE_SAMPLE_DATA shared database.
- TPC-DS consists of 7 fact tables and 17 dimensions
- *scale factor 10TB and 100TB*
- The largest table, STORE_SALES, contains nearly 300 billion rows, and the fact tables contain over 560 billion rows in total.

Getting Started with Data Sharing

- Data Sharing is an extremely powerful, yet easy-to-use feature. You can get started as a data provider in just a few simple steps.
- To perform the tasks described in this topic, you must use the ACCOUNTADMIN role

Step 1: Create a Share

```
CREATE [ OR REPLACE ] SHARE <name> [ COMMENT = '<string_literal>' ]
```

Creates a new, ***empty*** share. Once the share is created, you can include a database and objects from the database (schemas, tables, and views) in the share using the GRANT <privilege> … TO SHARE command. You can then use ALTER SHARE to add one or more accounts to the share.

Step 2: Add Objects to the Share by Granting Privileges

Use GRANT <privilege> … TO SHARE to grant the following object privileges to the share:

- USAGE privilege on the database you wish to share.
- USAGE privilege on each database schema containing the objects you wish to share.
- SELECT privilege for sharing specific objects (tables, secure views, and secure UDFs) in each shared schema.

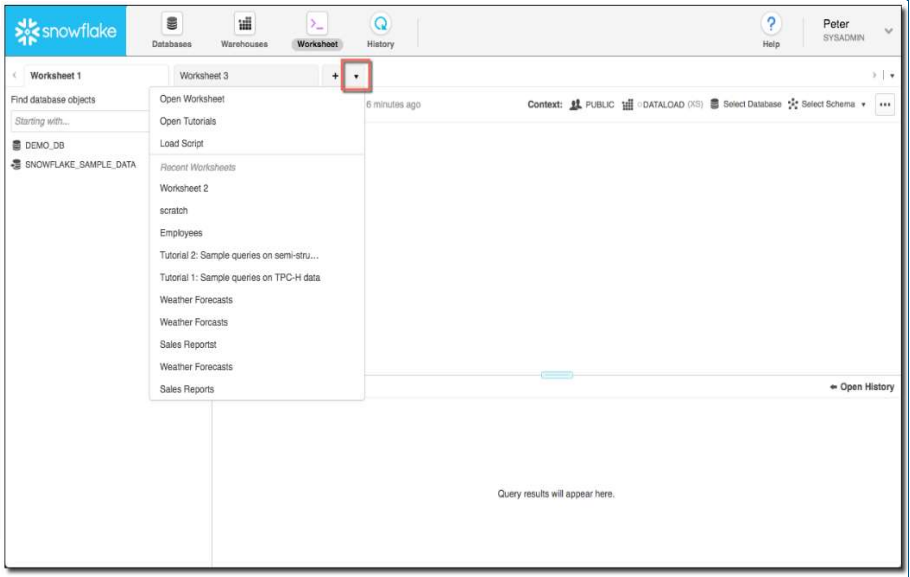
Optionally use SHOW GRANTS to view the object grants for the share.

Step 3: Add One or More Accounts to the Share

Use ALTER SHARE to add one or more accounts to the share.
To review the accounts added to the share, you can use SHOW GRANTS.

Example:

```
use role accountadmin;

create share sales_s;

grant usage on database sales_db to share sales_s;
grant usage on schema sales_db.aggregates_eula to share sales_s;
grant select on table sales_db.aggregates_eula.aggregate_1 to share sales_s;

show grants to share sales_s;

alter share sales_s add accounts=consumer_account1, consumer_account2;

show grants of share sales_s;
```

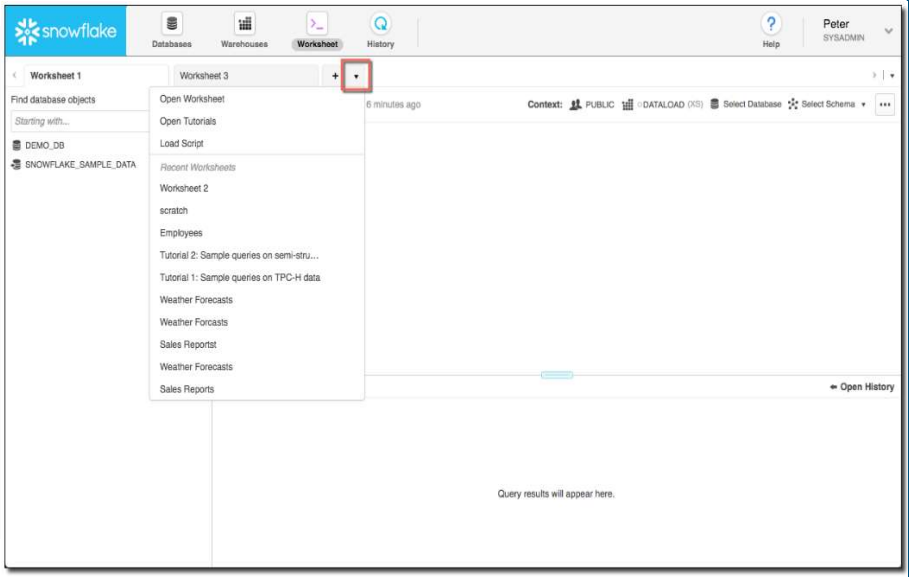


Step 3: Add One or More Accounts to the Share

Use ALTER SHARE to add one or more accounts to the share.
To review the accounts added to the share, you can use SHOW GRANTS.

Example:

```
use role accountadmin;

create share sales_s;

grant usage on database sales_db to share sales_s;
grant usage on schema sales_db.aggregates_eula to share sales_s;
grant select on table sales_db.aggregates_eula.aggregate_1 to share sales_s;

show grants to share sales_s;

alter share sales_s add accounts=consumer_account1, consumer_account2;

show grants of share sales_s;
```



Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses -- Covered**
  - ✓ **Understanding Databases and Storage -- Covered**
  - ✓ **Load Data into Snowflake -- Covered**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake-- Covered**
  - ✓ **Snowflake Internal Stage**
  - ✓ **Connecting to Snowflake**
  - ✓ **Query Data in Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**

- **Hands-On**

Snowflake Stage

Types of Stages

- User Stages
- Table Stages
- Internal Named Stages

User Stages

- Multiple users cannot access to the files.

Table Stages

- This option is not appropriate if you need to copy the data in the files into multiple tables.

Internal Named Stages

- Users with the appropriate privileges on the stage can load data into any table.
- Ownership of the stage can be transferred to another role, and privileges granted to use the stage can be modified to add or remove roles.
- When you create a stage, you must explicitly grant privileges on the stage to one or more roles before users use the stage.
- Named internal stages are optional but ***recommended*** when you plan regular data loads that could involve multiple users and/or tables.

Choosing a Stage for Local Files

Creating a Stage

```
create or replace stage my_stage file_format = my_csv_format;
```

Staging the Data Files

The following example uploads a file named `data.csv` in the `/data` directory on your local machine to a named internal stage called `my_stage`

Linux or Mac OS

```
put file:///data/data.csv @my_stage;
```

Windows

```
put file://C:\temp\contacts*.csv @my_stage;
```

Listing Staged Data Files

```
list @my_stage;
```


Snowflake Stage

Load data from your staged files into the target table.

```
copy into mytable from @my_stage;
```

Validating Your Data

- Before loading your data, you can validate that the data in the uploaded files will load correctly.
- To validate data in an uploaded file, execute COPY INTO <table> in validation mode using the VALIDATION_MODE parameter. The VALIDATION_MODE parameter returns any errors that it encounters in a file. You can then modify the data in the file to ensure it loads without error.
- In addition, the ON_ERROR copy option for the COPY INTO <table> command indicates what action to perform if errors are encountered in a file during loading.

Migrate your Data to Snowflake

- ✓ **Migrate to Snowflake**
  - Netezza Migration Guide Reference Guide
  - Oracle Migration Guide Reference Guide
  - Teradata Migration Guide Reference Guide

Supported Partners

Data Integration:

Alooma

Fivetran

Stitch

Snowflake Academy:

https://support.snowflake.net/s/get-started

Agenda

- **Get started with Snowflake**
  - ✓ **Snowflake Architecture Overview -- Covered**
  - ✓ **Snowflake User Interface -- Covered**
  - ✓ **Snowflake Administration -- Covered**
  - ✓ **Snowflake Security features -- Covered**
- **Learn how to use Snowflake:**
  - ✓ **Understanding Virtual Warehouses -- Covered**
  - ✓ **Understanding Databases and Storage -- Covered**
  - ✓ **Load Data into Snowflake -- Covered**
- **Migrate your Data to Snowflake:**
  - ✓ **Share Data in Snowflake-- Covered**
  - ✓ **Snowflake Internal Stage-- Covered**
  - ✓ **Connecting to Snowflake – Get into snowflake a/c.**
  - ✓ **Query Data in Snowflake**
- **Demo:**
  - ✓ **Sample Python scripts and implementation**
  - ✓ **Extract, Load, Transformations through script**
  - **Hands-On**

Query Data in Snowflake

- General DML
- Data Loading / Unloading DML
- File Staging Commands (for Data Loading / Unloading)

SELECT Query

```
select last_name from employee_table where employee_id = 101;
```

```
select department_name, last_name, first_name
from employee_table
inner join department_table on employee_table.department_id =
department_table.department_id
order by department_name, last_name, first_name;
```

```
-- Assuming that the radius of the circle is 2.0.
select pi() * 2.0 * 2.0 as area_of_circle; -- or

select pi() * pow(2.0, 2) as area_of_circle;
```

Querying Data in Staged Files

Snowflake supports using standard SQL to query data files located in an internal (i.e. Snowflake) stage or *named* external (i.e S3 or Azure) stage. This can be useful for inspecting/viewing the contents of the staged files, particularly before loading or after unloading data.

```
SELECT [<alias>.]$<file_col_num>[.<element>] [ , [<alias>.]$<file_col_num>[.<element>] , ... ] FROM { <internal_location> | <external_location>
} [ ( FILE_FORMAT => <named_file_format> ) ] [ <alias> ]
```


To stage and query the files:

```
-- Create a file format
create or replace file format myformat type = 'csv' field_delimiter = '|';

-- Create an internal stage
create or replace stage mystage1;
-- Stage the data files
put file:///tmp/data*.csv @mystage1;

-- Query the filename and row number metadata columns and the regular data columns in the staged file -- Note that the
table alias is provided to make the statement easier to read and is not required
select t.$1, t.$2 from @mystage1 (file_format => myformat) t;
```

```
+----+----+
| $1 | $2 |
|----+----|
| a  | b  |
| c  | d  |
| e  | f  |
| g  | h  |
+----+----+
```

```
select t.$1, t.$2 from @mystage1 t;
```

```
+-----+------+
| $1  | $2   |
|-----+------|
| a|b | NULL |
| c|d | NULL |
| e|f | NULL |
| g|h | NULL |
+-----+------+
```


General DML
INSERT
INSERT (multi-table)
MERGE
UPDATE
DELETE
TRUNCATE TABLE

INSERT

Updates a table by inserting one or more rows into the table. The values inserted into each column in the table can be explicitly-specified or the results of a query.

```
INSERT [ OVERWRITE ] INTO <target_table> [ ( <target_col_name> [ , ... ] ) ] { { VALUES ( { <value> | DEFAULT | NULL } [ , ... ] ) [ , ( ... ) ]
} | <query> }
```

Single Row Insert Using a Query

```
insert into mytable select to_date('2013-05-08T23:39:20.123'), to_timestamp('2013-05-08T23:39:20.123'), to_timestamp('2013-05-08T23:39:20.123');
```

Multi-row Insert Using Explicitly-specified Values

```
insert into employees values ('Lysandra','Reeves','1-212-759-3751','New York',10018), ('Michael','Arnett','1-650-230-8467','San
Francisco',94116);
```

Multi-row Insert Using Query

```
insert into employees(first_name, last_name, workphone, city,postal_code) select contractor_first,contractor_last,worknum,null,zip_code from
contractors where contains(worknum,'650');


insert into emp (id,first_name,last_name,city,postal_code,ph) select a.id,a.first_name,a.last_name,a.city,a.postal_code,b.ph from emp_addr a
inner join emp_ph b on a.id = b.id;
```


Multi-row Insert for JSON Data

Insert two JSON objects into a VARIANT column in a table:

```
insert into prospects
select parse_json(column1)
from values
('{ "_id": "57a37f7d9e2b478c2d8a608b",
    "name":
            { "first": "Lydia",
            "last": "Williamson" },
            "company": "Miralinz",
            "email": "lydia.williamson@miralinz.info",
            "phone": "+1 (914) 486-2525",
            "address": "268 Havens Place, Dunbar, Rhode Island, 7725" }') ,

('{ "_id": "57a37f7d622a2b1f90698c01",
    "name": {
            "first": "Denise",
            "last": "Holloway" },
            "company": "DIGIGEN",
            "email": "denise.holloway@digigen.net",
            "phone": "+1 (979) 587-3021",
            "address": "441 Dover Street, Ada, New Mexico, 5922" }');
```


Insert Using Overwrite

Insert using overwrite to rebuild `sf_employees` table from `employees` after new records were added to `employees`:

```
select * from employees;

+------------+-----------+----------------+---------------+-------------+
| FIRST_NAME | LAST_NAME | WORKPHONE      | CITY          | POSTAL_CODE |
|------------+-----------+----------------+---------------+-------------|
| May        | Franklin  | 1-650-249-5198 | San Francisco | 94115       |
| Gillian    | Patterson | 1-650-859-3954 | San Francisco | 94115       |
| Lysandra   | Reeves    | 1-212-759-3751 | New York      | 10018       |
| Michael    | Arnett    | 1-650-230-8467 | San Francisco | 94116       |
+------------+-----------+----------------+---------------+-------------+

insert overwrite into sf_employees
  select * from employees
  where city = 'San Francisco';

select * from sf_employees;

+------------+-----------+----------------+---------------+-------------+
| FIRST_NAME | LAST_NAME | WORKPHONE      | CITY          | POSTAL_CODE |
|------------+-----------+----------------+---------------+-------------|
| May        | Franklin  | 1-650-249-5198 | San Francisco | 94115       |
| Gillian    | Patterson | 1-650-859-3954 | San Francisco | 94115       |
| Michael    | Arnett    | 1-650-230-8467 | San Francisco | 94116       |
+------------+-----------+----------------+---------------+-------------+
```

```
insert into employees
  values ('Laurel', 'Slater', '1-650-633-4495', 'San Francisco', '94112');

insert overwrite into sf_employees
  select * from employees
  where city = 'San Francisco';
```

```
select * from sf_employees;

+------------+-----------+----------------+---------------+-------------+
| FIRST_NAME | LAST_NAME | WORKPHONE      | CITY          | POSTAL_CODE |
|------------+-----------+----------------+---------------+-------------|
| May        | Franklin  | 1-650-249-5198 | San Francisco | 94115       |
| Gillian    | Patterson | 1-650-859-3954 | San Francisco | 94115       |
| Michael    | Arnett    | 1-650-230-8467 | San Francisco | 94116       |
| Laurel     | Slater    | 1-650-633-4495 | San Francisco | 94112       |
+------------+-----------+----------------+---------------+-------------+
```


INSERT (multi-table)

Updates multiple tables by inserting one or more rows with column values (from a query) into the tables. Supports both unconditional and conditional inserts.

```
-- Unconditional multi-table insert
INSERT [ OVERWRITE ] ALL intoClause [ ... ] <subquery>
-- Conditional multi-table insert
INSERT [ OVERWRITE ] { FIRST | ALL } { WHEN <condition> THEN intoClause [ ... ] } [ ... ] [ ELSE intoClause ] <subquery>
```

Where:

```
intoClause ::= INTO <target_table> [ ( <target_col_name> [ , ... ] ) ] [ VALUES ( { <source_col_name> | DEFAULT | NULL } [ , ... ] ) ]
```

Unconditional Multi-table Inserts

```
insert all
  into t1
  into t1 (c1, c2, c3) values (n2, n1, default)
  into t2 (c1, c2, c3)
  into t2 values (n3, n2, n1)
select n1, n2, n3 from src;

-- If t1 and t2 need to be truncated before inserting, OVERWRITE must be specified
insert overwrite all
  into t1
  into t1 (c1, c2, c3) values (n2, n1, default)
  into t2 (c1, c2, c3)
  into t2 values (n3, n2, n1)
select n1, n2, n3 from src;
```

Conditional Multi-table Inserts

```
insert all
  when n1 > 100 then
    into t1
  when n1 > 10 then
    into t1
    into t2
  else
    into t2
select n1 from src;
```


MERGE

Inserts, updates, and deletes values in a table based on values in a second table or a subquery. This can be useful if the second table is a change log that contains new rows (to be inserted), modified rows (to be updated), and/or marked rows (to be deleted) in the target table.

The command supports semantics for handling the following cases:
- Values that match (for updates and deletes).
- Values that do not match (for inserts).

```
MERGE INTO <target_table> USING <source> ON <join_expr> { matchedClause | notMatchedClause } [ ... ]
```

Where:

```
matchedClause ::= WHEN MATCHED [ AND <case_predicate> ] THEN { UPDATE SET { <col_name> = <expr> } | DELETE } [ ... ]
notMatchedClause ::= WHEN NOT MATCHED [ AND <case_predicate> ] THEN INSERT [ ( <col_name> [ , ... ] ) ] VALUES ( <expr> [ , ... ] )
```

Perform a basic merge:

```
merge into t1 using t2 on t1.t1key = t2.t2key
           when matched and t2.marked = 1 then delete
           when matched and t2.isnewstatus = 1 then update set val = t2.newval, status = t2.newstatus
           when not matched then insert (val, status) values (t2.newval, t2.newstatus);
```


UPDATE, DELETE & TRUNCATE

```
UPDATE <target_table> SET <col_name> = <value> [ , <col_name> = <value> , ... ] [ FROM <additional_tables> ] [ WHERE <condition> ]
```

```
update t1 set t1.number_column = t1.number_column + t2.number_column, t1.text_column = 'ASDF' from t2 where
t1.key_column = t2.t1_key and t1.number_column < 10;
```

DELETE

Unlike TRUNCATE TABLE, this command does ***not*** delete the external file load history. If you delete rows loaded into the table from a staged file, you cannot load the data from that file again unless you modify the file and stage it again.

```
DELETE FROM <table_name> [ USING <additional_tables> ] [ WHERE <condition_query> ]
```

```
delete from tab1 using tab2 where tab1.key_column = tab2.tab1_key and tab2.number_column < 10;
```

TRUNCATE

Removes all rows from a table but leaves the table intact (including all privileges and constraints on the table). Also deletes the load metadata for the table, which allows the same files to be loaded into the table again after the command completes.
Note that this is ***different*** from DROP TABLE, which removes the table from the system but retains a version of the table (along with its load history) so that they can be recovered.

```
TRUNCATE [ TABLE ] <name>
```

```
-- truncate the table
truncate table if exists temp;
```


Data Loading / Unloading DML

Commands for bulk copying data into and out of Snowflake tables:
- COPY INTO <table> (loading/importing data)
- COPY INTO <location> (unloading/exporting data)

COPY INTO <table> (loading/importing data)

```
-- S3 bucket copy into mytable from 's3://mybucket/./../a.csv';

-- Azure container copy into mytable from 'azure://myaccount.blob.core.windows.net/mycontainer/./../a.csv';
```

COPY INTO <location> (unloading/exporting data)

```
-- S3 bucket copy into 's3://mybucket/./../a.csv' from mytable;

-- Azure container copy into 'azure://myaccount.blob.core.windows.net/mycontainer/./../a.csv' from mytable;
```

File Staging Commands (for Data Loading / Unloading)

These commands do not perform any actual DML, but are used to stage and manage files stored in Snowflake locations (named internal stages, table stages, and user stages), for the purpose of loading and unloading data:

- PUT FROM LOCAL TO INTERNAL STAGE

```
put file:///tmp/data/mydata.csv @my_int_stage;
```

- GET FROM INTERNAL STAGE TO local

```
get @%mytable file:///tmp/data/;
```

- LIST (can also be used with named external stages)

```
list @%mytable;
```

- REMOVE

```
remove @%orderstiny_ext;
```


Migrate your Data to Snowflake

- ✓ **Migrate to Snowflake**
  - Netezza Migration Guide Reference Guide
  - Oracle Migration Guide Reference Guide
  - Teradata Migration Guide Reference Guide

Supported Partners

Data Integration:

Alooma

Fivetran

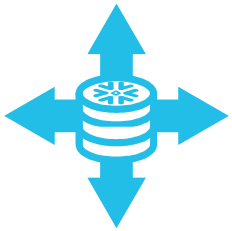
Stitch

Snowflake Academy:

https://support.snowflake.net/s/get-started

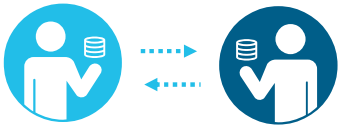
Snowflake's differentiating technology

Single place for data (structured + semi-structured)

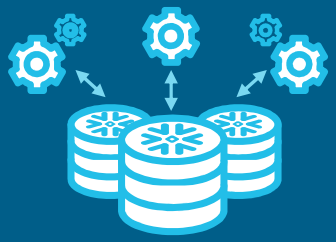


Instant, unlimited scalability

Minimal management



Instant, live data sharing



Unique architecture: Multi-cluster, shared data

Competition

Category	Examples	Similarities	Snowflake Differentiators
On-premises EDW	Teradata, Vertica, Netezza, Oracle Exadata	• MPP architecture • Native SQL	• Instant scalability • Separation of compute and storage • No need for data distribution
Cloud EDW	Redshift, Azure SQL DW, Teradata IntelliCloud, Vertica Eon	• No physical hardware to manage • Native SQL	• Concurrency • Automatic failover and disaster recovery • Built for the cloud
Hadoop	Cloudera, Hortonworks, MapR	• Parallelism • Support for semi-structured data	• No need to manage files • Native SQL (including on semi-structured)
Data Engines	Big Query, Spark, Presto	• Separation of Storage and compute • Parallelism	• No need to manage files • Native SQL

Query execution

1. **Query received by Snowflake**
   Sent via standard ODBC, JDBC, or web UI interfaces

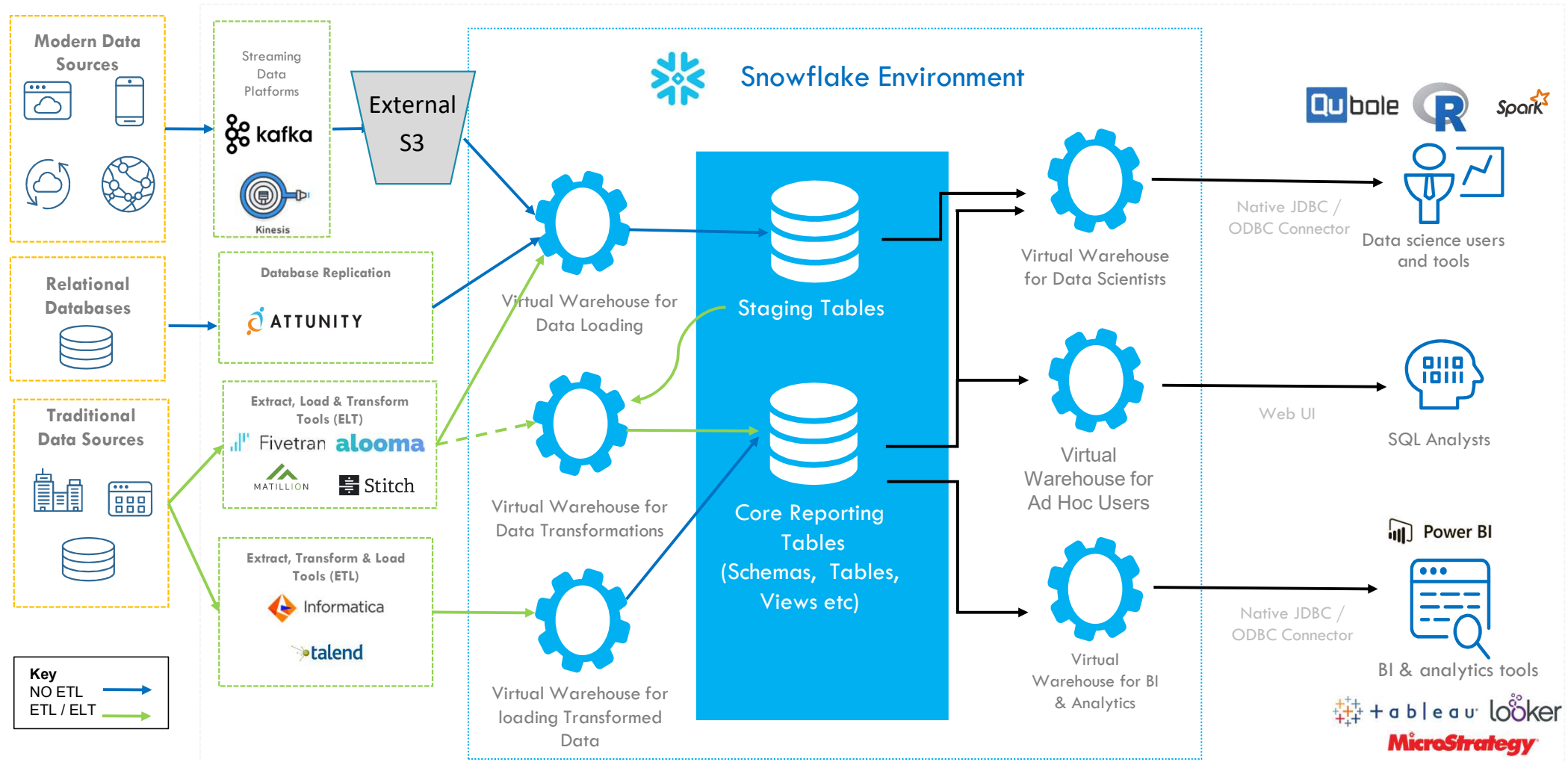
2. **Result cache lookup**
   If the query matches an entry in the result cache then the result is returned immediately

3. **Planner and optimizer process query**
   Prune and filter, then use metadata to identify exact data to be processed (or retrieved from result cache)

4. **Virtual warehouse processing**
   Virtual warehouse scans only needed data from local SSD cache or Amazon S3, processes, and returns to cloud services

5. **Result set return**
   Final result processed, stored in cache for future use, and returned to client

Data Warehouse Modernization using Snowflake



Q&A

Thank you

People matter, results count.

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.capgemini.com