

# Snowflake Core



INNOVATION  
SOFTWARE  
EMPOWER. INNOVATE.

Develop a passion for learning.

# COURSE OBJECTIVES

Attendees will leave

- understanding the Snowflake architecture
- knowing how to load and transform data
- knowing how to evaluate query constructs, DDL and DML Operations
- managing identity and access
- learning best practices for working with data
- employing Snowflake's method for continuous data protection
- understanding Snowflake security
- preparing for the Snowflake Core certification

# Agility



## Data Warehousing Overview

- Data warehousing evolution
- Cloud data warehousing
- Adapting to increasing demands for data access
- Adjusting to how data is created and used today

## Architecture and Overview

- Technical Overview
- Cloud Services Layer
- Compute Layer
- Storage Layer
- Optimization
- Metadata
- Governance

## Data Movement

- Data Loading
- Data Unloading
- Best Practices
- Data Sharing
- Data Lineage
- Snowpipe



## Objects and Commands

- Query Constructs
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Local only resources

## SQL Support for Data Analysis

- SQL Support and Query Best Practices
- SQL Analytic Functions
- High Performing Estimation Functions
- UDF and Stored Procedure
- Demo Query Profile
- Transactions

## Managing Security

- Data Encryption
- Authentication
- Role-Based Access Control
- External Validation

## Data Modeling

- Schema on Read vs Schema on Write
- Transforming JSON to RDBMS
- JSON to 3NF
- JSON to Data Vault Model



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# Observability



## Semi-structured data

- Working with semi-structured data
- Queries
- Data Optimization

## Caching

- Caching Features
- Performance Improvements
- Cost Optimization

## Clients and Ecosystem

- Clients
- Connectors
- SnowSQL

## Modern Data Sharing

- Data Analytics as a Service
- External Data Sharing
- Monetizing Data
- Sharing Data Sets
- Collaboration



# Launch



## Security

- Continuous Data Protection
- Time Travel
- Cloning
- Fail-Safe

## Performance and Concurrency

- Query Profile
- Micro-Partitions
- Data Clustering
- Scaling a Virtual Warehouse

## Account and Resources

### Management and Monitoring

- System Resource Usage and Billing
- Managing Virtual Warehouses
- Workload Independence and Segmentation
- Resource Monitors
- Information Schema and Account Usage

## Best Practices Highlights

- Infrastructure
- Network
- Security
- Data Ingestion
- Performance
- Catalog (Collibra/ 3rd Party Tools)
- Architecture
- BigData



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# THE TRAINING DILEMMA

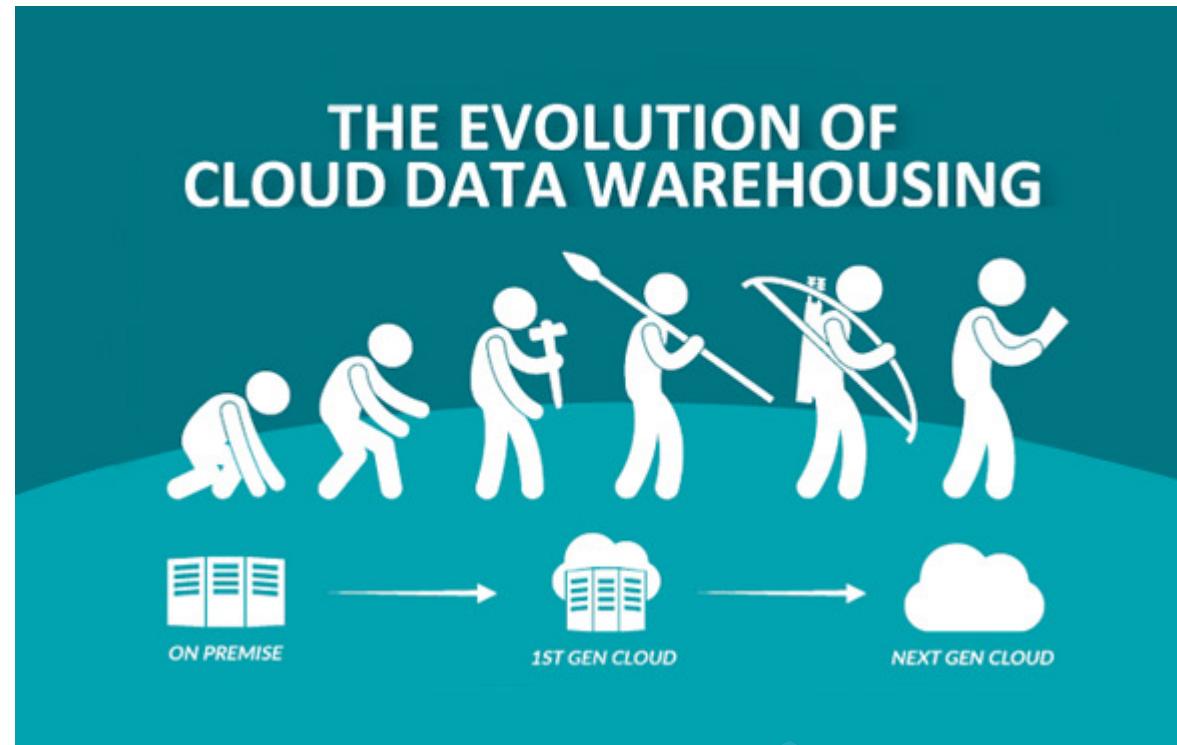


INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# Data Warehouse Overview



# DATA WAREHOUSING EVOLUTION



# DATA WAREHOUSE CHARACTERISTICS

Warehouses are required for queries, as well as all DML operations, including loading data into tables. A warehouse is defined by its size, as well as the 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.

# **SNOWFLAKE DATA WAREHOUSE**

A **data warehouse** is a relational database that is designed for analytical rather than transactional work.

It collects and aggregates data from one or many sources so it can be analyzed to produce business insights.

It serves as a federated repository for all or certain data sets collected by a business's operational systems. **Snowflake is a cloud data warehouse**

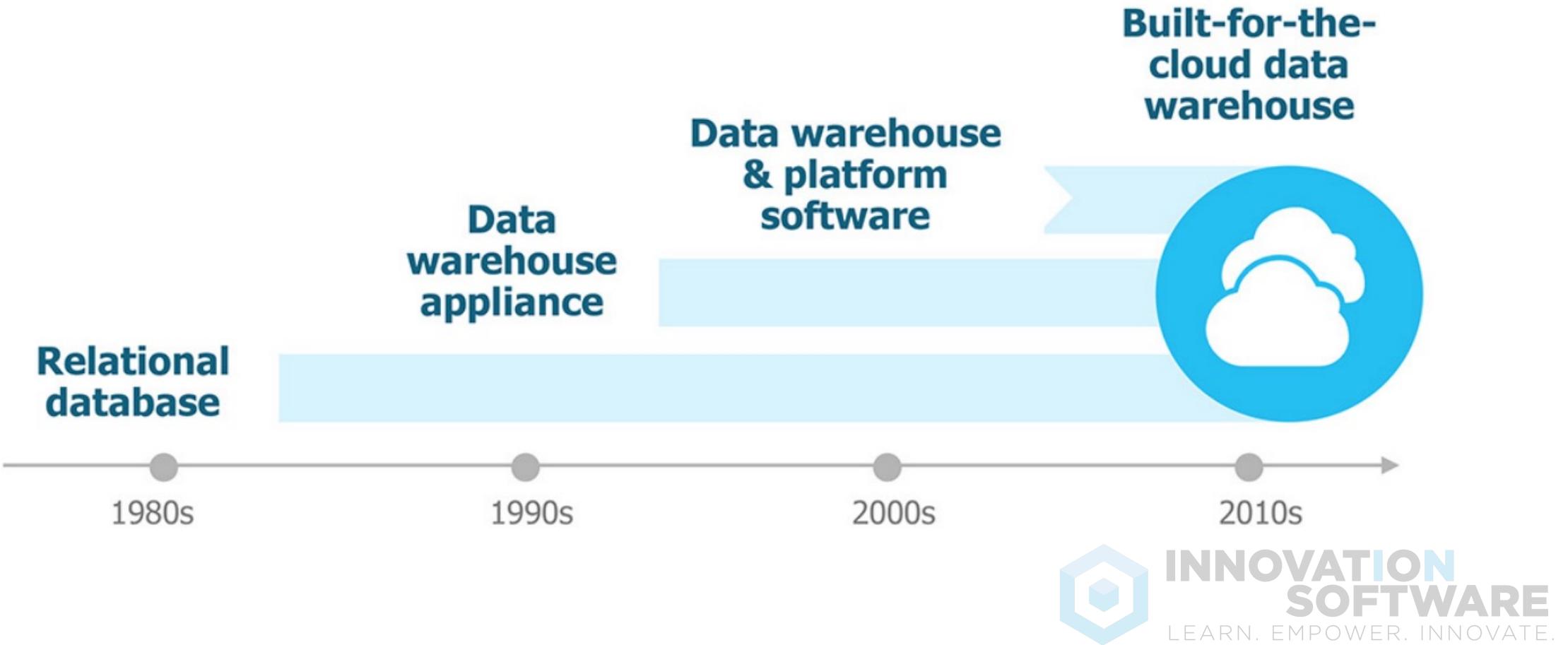


# SNOWFLAKE DATA WAREHOUSE

- Founded in 2012 – product launched in 2015
- Runs on AWS, GCP & Azure
- Raised \$1.4B prior to IPO
- Snowflake had 4,532 customers as of May'2021
- Completed IPO in September
- Largest software IPO
- Currently valued at ~75B\$
- ~3,300 employees

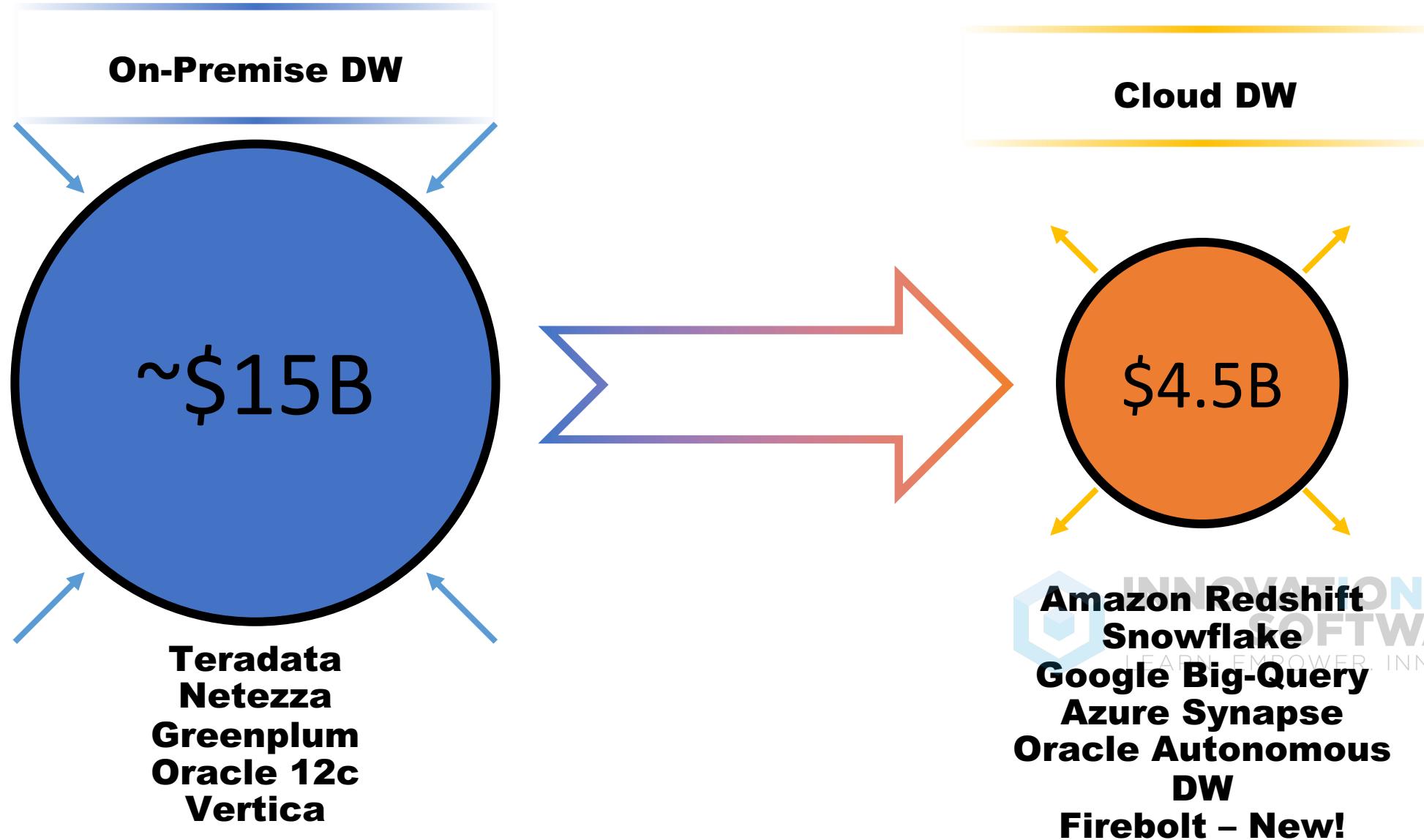


# CLOUD DATA WAREHOUSING



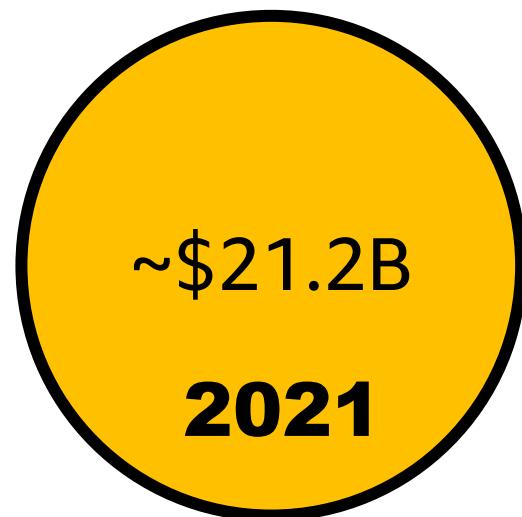
**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# CLOUD DATA WAREHOUSING



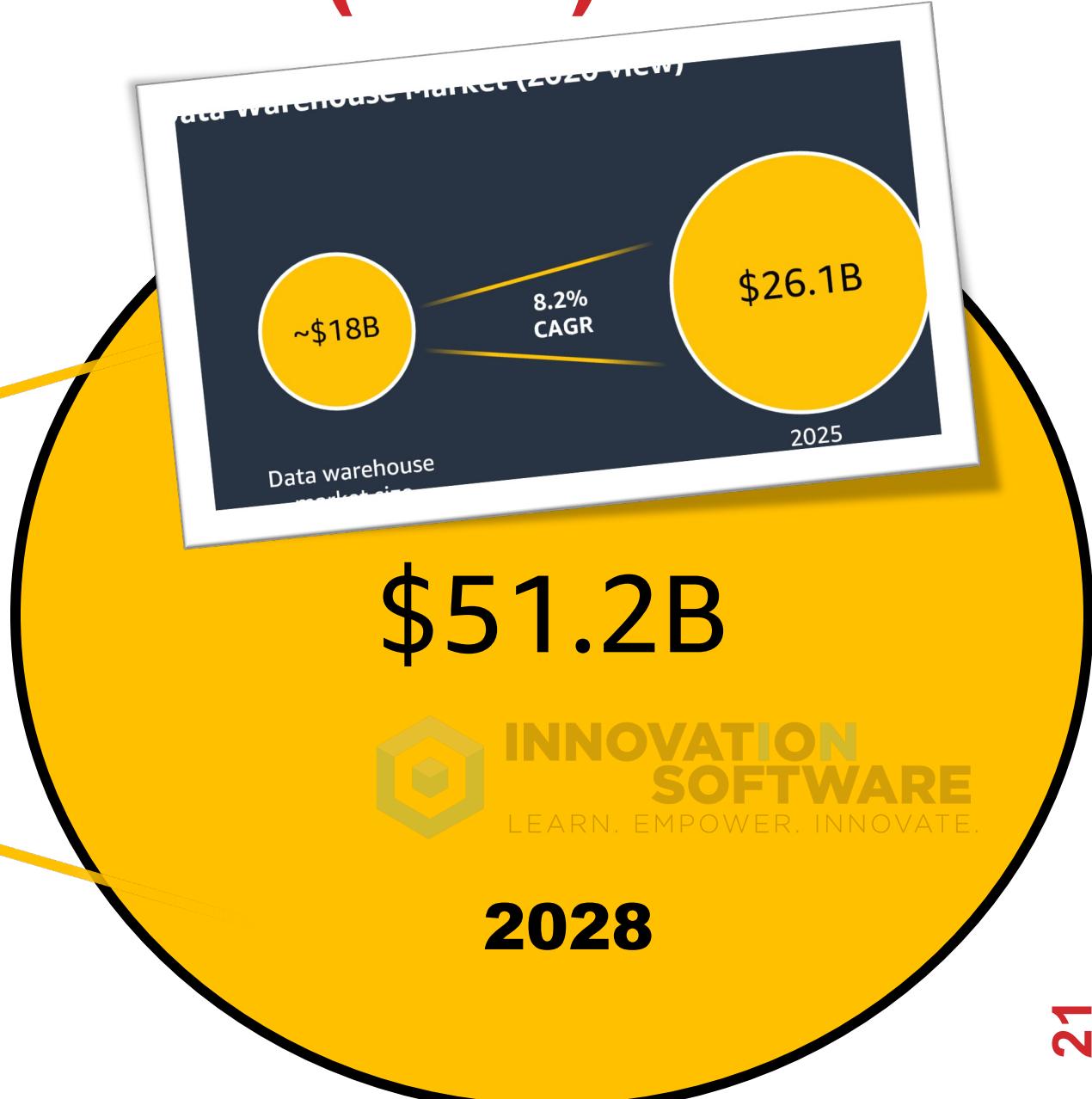
# DATA WAREHOUSE MARKET (2021)

Source: Allied Market Research



**Data  
warehouse  
market size**

**10.7%  
CAGR**



# ADAPTING TO INCREASING DEMANDS FOR DATA ACCESS

**5.8B**

endpoints in use in  
2020 at 21% growth

- Gartner

**\$200B**

data center market in  
2021 at 6% growth

- Gartner

**\$335B**

global public cloud  
market by 2022 at  
12% growth

- Gartner

**40%**

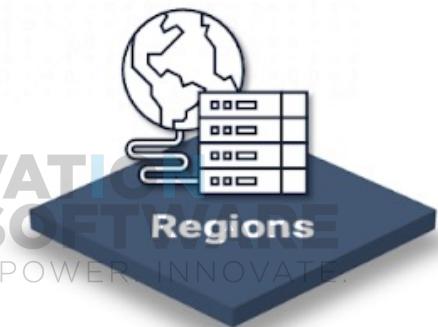
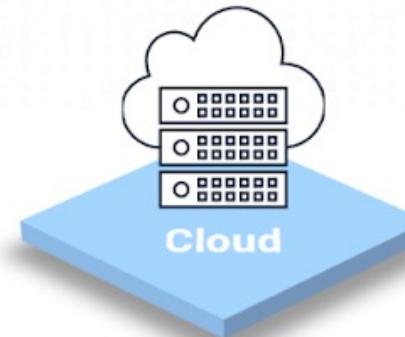
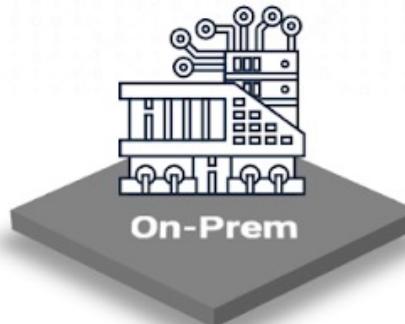
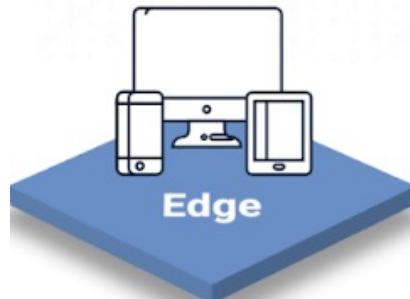
of companies in the  
cloud will use 2+  
providers

- Gartner

**70%**

think cloud is more  
complex due to local  
privacy regulations

- Privacera

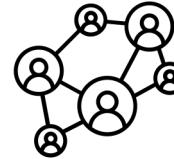


INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

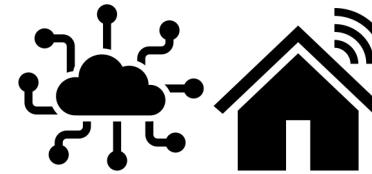
# ADJUSTING TO HOW DATA IS CREATED AND USED TODAY



Commerce Data



Social Media



Sensors/IoT



Mobile Devices



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# WHAT CUSTOMERS DO WITH CLOUD DW

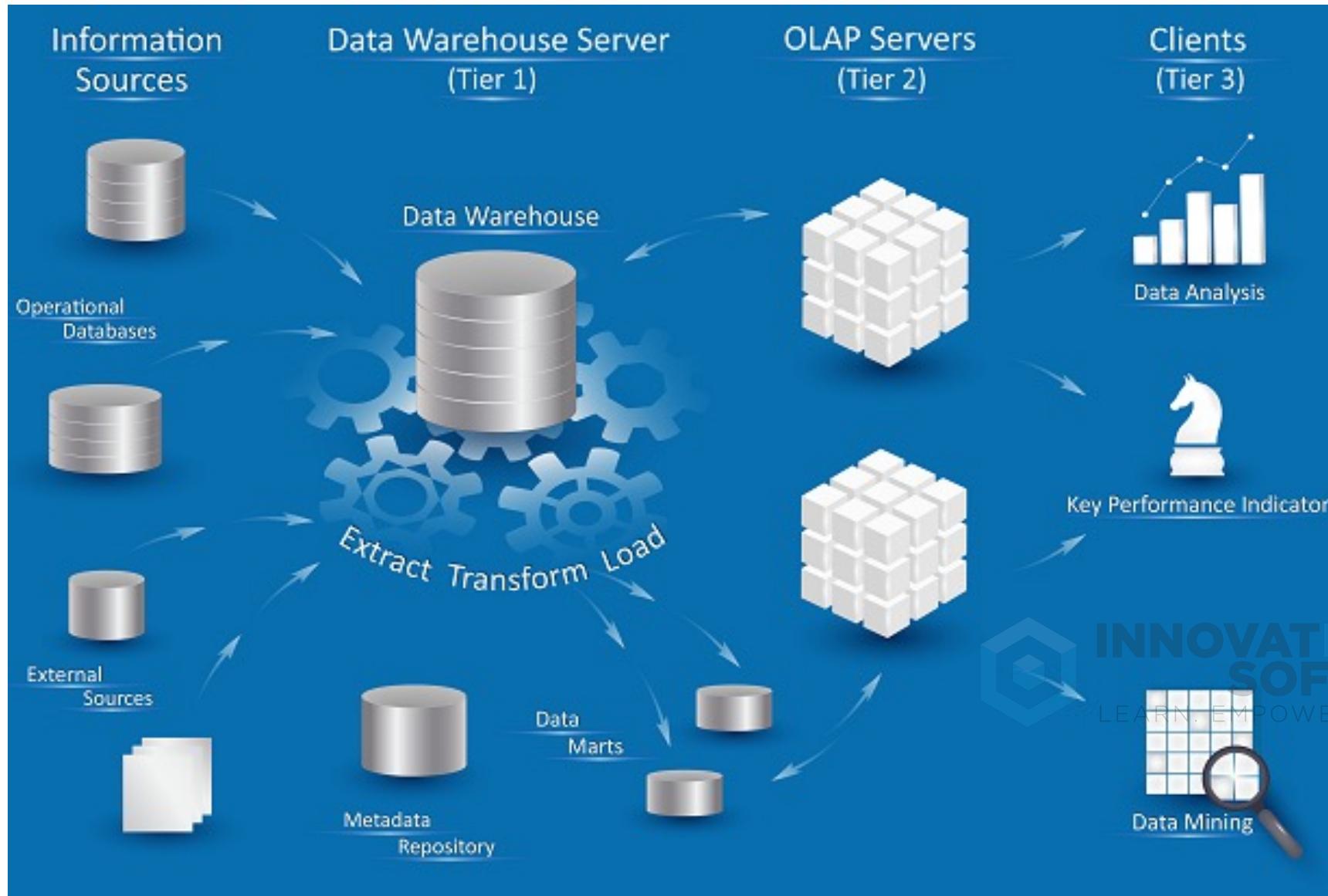
- Gaming company **replaced Hadoop + SQL database with Snowflake**
- Consumer retailer modernizing DW by **replacing the legacy appliance with Snowflake**
- Market research company **consolidated data marts** to reduce costs and data silos
- Mobile analytics company **shares live data** with clients

# DATA WAREHOUSE SELECTION CRITERIA

When you're in the market for a **data warehouse**, a **checklist of criteria** will help determine which alternative best meets your needs.

- Supports Existing Skills, Tools, and Expertise
- Saves Your Organization Money
- Provides Data Resiliency and Recovery
- Secures Data at Rest and in Transit
- Streamlining the data pipeline
- Optimizes Your Time to Value

# BUSINESS INTELLIGENCE



INNOVATION  
SOFTWARE

LEARN. EMPOWER. INNOVATE.



# DATA WAREHOUSE VS. DATABASE

A data warehouse focuses on collecting data from multiple sources to facilitate broad access and analysis. They specialize in data aggregation and providing a longer view of an organization's data over time.

A data warehouse is optimized to store large volumes of historical data and enables fast and complex querying of that data. Standard operational databases focus on transactional functions such as real-time data updates for ongoing business processes.



# DATA WAREHOUSE KEY FUNCTIONS

Data warehousing has two key functions.

**First**, it serves as a **historical repository** for integrating the information and data that is needed by the business, which may come from a variety of different sources.

**Second**, it serves as a **query execution and processing engine** for that data, enabling end users to interact with the data that is stored in the database.



# COMPLEX QUERY EXECUTION ENGINE

Complex queries are very difficult to run without a temporary pause of database update operations. A frequently paused transactional database will inevitably lead to data errors and gaps.

Therefore, a data warehouse serves as a **separate platform for aggregation** across multiple sources and then for analytics tasks across those diverse sources. This separation of roles allows databases to remain focused on purely transactional jobs without interruption.



# DIGITAL DATA SUPPLY CHAIN

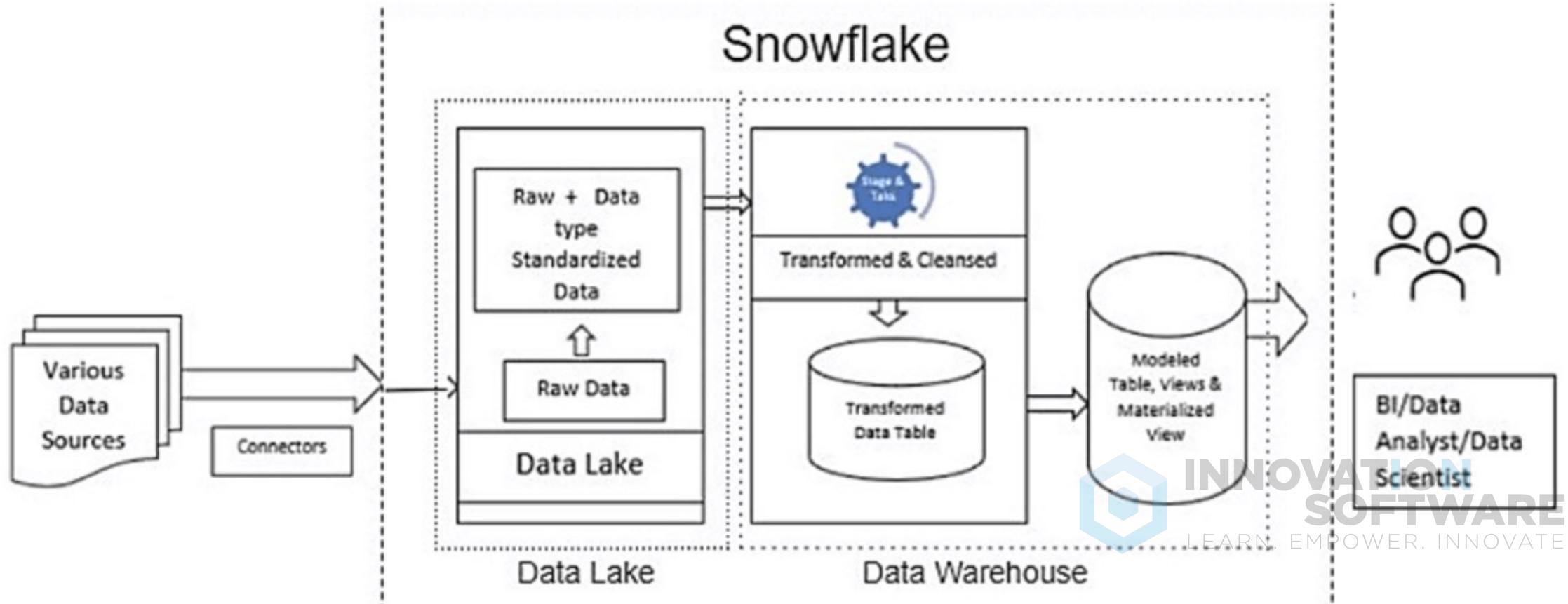


# DATA WAREHOUSE COMPONENTS

A data warehouse usually consists of data sources from operational and transactional systems (ERP, CRM, finance apps, IoT devices, mobile and online systems) as well as:

- A data staging area for aggregation and cleaning
- A presentation/access area where data is warehoused for analytics (querying, reporting) and sharing
- A range of data tool integrations or APIs (BI software, ingestion and ETL tools, etc.).

# DATA WAREHOUSE, LAKE, LAKEHOUSE



# GETTING STARTED WITH CDW

Snowflake gives us a free Trial account with 400 usage credits to explore Cloud Data Warehouse.

Choose your Snowflake edition\*

Standard

A strong balance between features, level of support, and cost.

Enterprise

Standard plus 90-day time travel, multi-cluster warehouses, and materialized views.

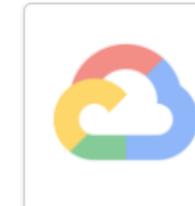
Business Critical

Enterprise plus enhanced security, data protection, and database failover/fallback.

Choose your cloud provider\*



Amazon Web Services



Google Cloud Platform



Microsoft Azure

US East (Northern Virginia)



**INNOVATION SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

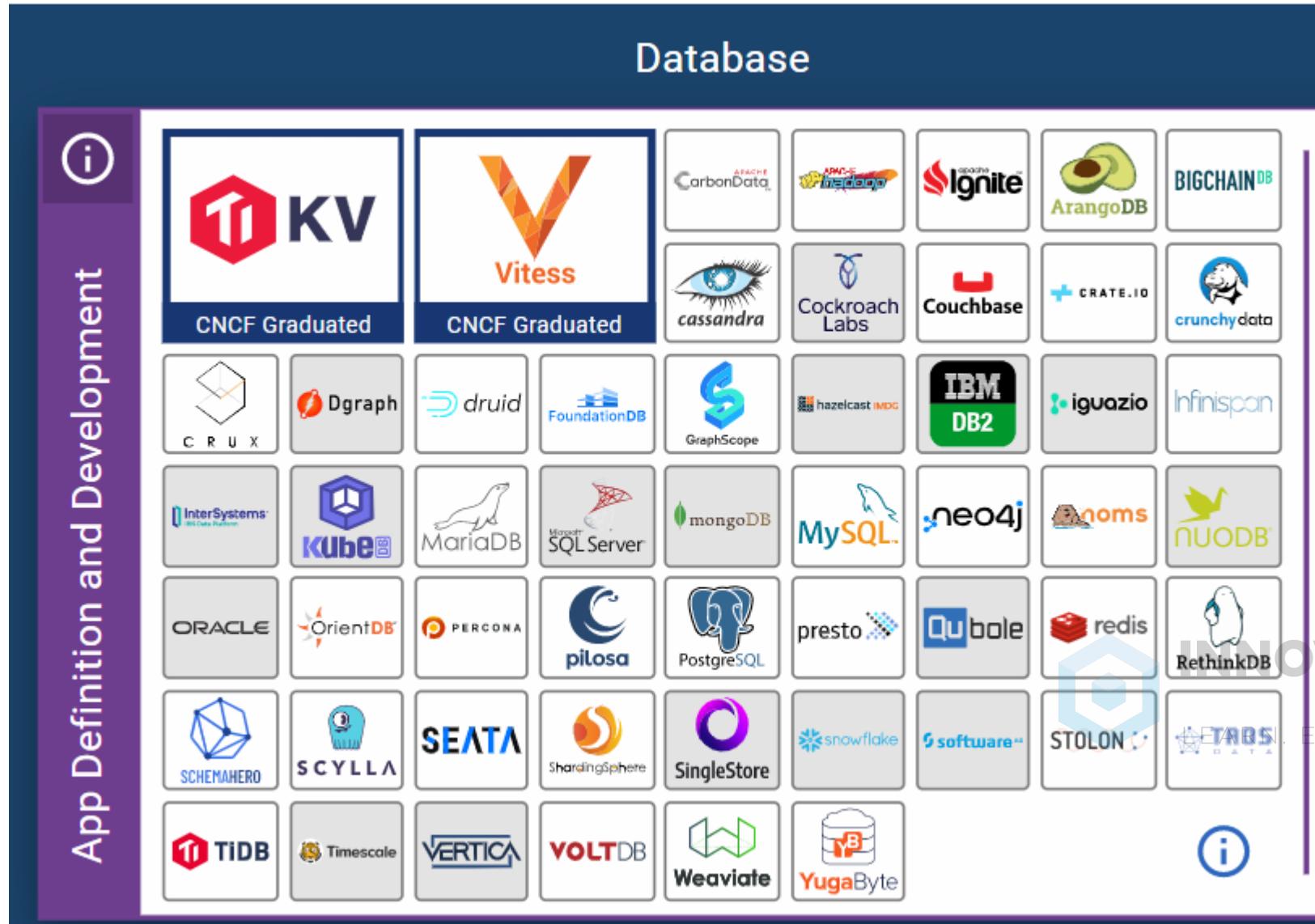
- Check here to indicate that you have read and agree to the terms of the Snowflake Self Service On Demand Terms.

GET STARTED

# SNOWFLAKE CDW JUMPSTART

- Snowflake has free courses to get started.
- Snowflake has a simplified UI designed to help learners focus effectively during the learning process.
- Snowflake does not require an up-front purchase. Instead, it has free trial credits and then converts to a pay-as-you-go model.

# SNOWFLAKE IN CLOUD-NATIVE CONTEXT



INNOVATION  
SOFTWARE  
EMPOWER. INNOVATE.

# SNOWFLAKE ACCOUNT

The screenshot shows the Snowflake Worksheet interface. At the top, there is a navigation bar with tabs for 'Activate Your Snowflake Account' (with an 'X'), 'Worksheet' (highlighted), and a '+' icon. Below the navigation bar is a URL bar showing the address 'lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet'. A promotional message 'Enjoy your free trial! Visit our documentation to...' is visible. The main menu includes icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (which is selected and highlighted in grey), and History.

On the left, there is a sidebar for 'New Worksheet' creation, a search bar for 'Find database objects' starting with 'Starting with...', and a list of databases: DEMO\_DB, SNOWFLAKE\_SAMPLE\_DATA, and UTIL\_DB.

A modal window titled 'Browse database objects' is open in the center. It contains the text: 'Use the object explorer to see databases, schemas, tables and views without having to leave your worksheet.' There is a blue hexagonal icon to the right of the text. At the bottom of the modal are three small dots and a 'NEXT' button.

INNOVATION SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWFLAKE ACCOUNT IDENTIFIERS

The account identifier for an account in your organization takes one of the following forms, depending on where and how the identifier is used:

**organization\_name-account\_name** (for most URLs and other general purpose usage)

**organization\_name\_account\_name** (for scenarios/features where hyphens are not supported in URLs)

**organization\_name.account\_name** (for SQL commands and operations)

For **trial accounts** the identifier is a randomized 8 character and number combination starting with 2-3 characters and the remainder numbers e.g. LVE098976



# SNOWFLAKE ACCOUNT IDENTIFIERS

The account identifier for an account in your organization takes one of the following forms, depending on where and how the identifier is used:

**organization\_name-account\_name** (for most URLs and other general purpose usage)

**organization\_name\_account\_name** (for scenarios/features where hyphens are not supported in URLs)

**organization\_name.account\_name** (for SQL commands and operations)

For **trial accounts** the identifier is a randomized 8 character and number combination starting with 2-3 characters and the remainder numbers e.g. LVE098976



# SNOWFLAKE ACCOUNT URLs

If you have accounts with the same name in different regions, the cloud and region names are prepended to the account name in the new URL format.

For example, if the organization name is ACME, and there are two accounts named TEST, one in the AWS US East 2 region and the other in the Azure West US 2 region, the URLs will look as follows:

Legacy account URL 1: <https://test.us-east-2.aws.snowflakecomputing.com>

Legacy account URL 2: <https://test.west-us-2.azure.snowflakecomputing.com>

New account URL 1: [https://acme-test\\_aws\\_us\\_east\\_2.snowflakecomputing.com](https://acme-test_aws_us_east_2.snowflakecomputing.com)

New account URL 2: [https://acme-test\\_azure\\_west\\_us\\_2.snowflakecomputing.com](https://acme-test_azure_west_us_2.snowflakecomputing.com)

# SNOWFLAKE WORKSHEET - CLASSIC

The screenshot shows the Snowflake Worksheet - Classic interface. At the top, there are two tabs: "Activate Your Snowflake Account" and "Worksheet - New Worksheet (1/1)". The URL in the address bar is "lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet". Below the tabs, there is a banner with the text "Enjoy your free trial! Visit our documentation". The navigation menu includes "Databases", "Shares", "Data Marketplace", "Warehouses", "Worksheets" (which is selected and highlighted in grey), and "History". On the left, a sidebar shows a list of databases: "DEMO\_DB", "INFORMATION\_SCHEMA", "PUBLIC" (with the note "No Tables or Views in this Schema"), and "SNOWFLAKE\_SAMPLE\_DATA". A "New Worksheet" button is visible. In the main workspace, a query is being typed into a text area:

```
1 use database snowflake_sample_data;
2
3
4
5
6 select cc_name, cc_companyname, cc mana
```

A tooltip above the "Run" button says "Run Query (Cmd/Ctrl + Return)". The "Run" button is blue with white text. To the right of the run button, there is a checkbox labeled "All Queries" and the text "Saved 27 seconds ago". The INNOVATION SOFTWARE logo is visible in the bottom right corner.

# SNOWFLAKE QUERY GETTING STARTED

The screenshot shows the Snowflake Worksheet interface. On the left, there's a sidebar with a tree view of database objects. The main area has a toolbar with icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (which is selected), and History. The top bar shows the URL <https://lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet>. A message at the top says "Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact our support team with any questions." The user is logged in as SUSANMARTIN with SYSADMIN privileges. A modal dialog box is open, asking "Do you want to run the following queries?" It contains the SQL command `use database snowflake_sample_data;`. There are "Copy SQL", "Cancel", and "Run" buttons. Below the dialog, it says "Query results will appear here."



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.  
← Open History

# QUERY RESULTS

Results Data Preview

✓ [Query ID](#) [SQL](#) 79ms  1 rows

Filter result... [!\[\]\(880294d3d5b36c2aa605b280dc09f6f7\_img.jpg\)](#) [Copy](#)

Row	status
1	Statement executed successfully.



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# QUALIFIED AND UNQUALIFIED QUERIES

Run  All Queries | Saved 19 seconds ago  SYSADMIN  COMPUTE\_WH (XS)

```
1 use database snowflake_sample_data;
2 use schema TPCDS_SF100TCL;
3
4 select cc_name,cc_manager from call_center;
5
```

Results Data Preview

✓ Query\_ID SQL 889ms  60 rows

Filter result...  

Row	CC_NAME	CC_MANAGER
1	NY Metro	Bob Belcher
2	Mid Atlantic	Felipe Perkins
3	Mid Atlantic	Mark Hightower



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWFLAKE WORKSHEET

Worksheets Object Tags Data Classification Row Access Load JSON to RDMBS install snowsql Load JSON to RDMBS JSO... + PREVIEW

Databases Worksheets

Pinned (0) No pinned objects

All Objects ...

SNOWFLAKE SNOWFLAKE\_SAMPLE\_DATA SNOWPIPE SNOWPIPE2 SNOWPIPE3 SNOWTEST

SNOWTEST.PUBLIC Settings

Latest Version Draft Q

```
1 USE ROLE ACCOUNTADMIN;
2
3 CREATE OR REPLACE DATABASE SNOWTEST;
4 CREATE OR REPLACE SCHEMA SNOWTEST.PUBLIC;
5
6 CREATE OR REPLACE TABLE home_sales (
7     json_column variant
8 );
9
10 CREATE OR REPLACE FILE FORMAT sf_tut_json_format
11     TYPE = JSON;
12
13 CREATE OR REPLACE TEMPORARY STAGE sf_tut_stage
14     FILE_FORMAT = sf_tut_json_format;
15
16 PUT 'file:///G:/Shared drives/IDSTS Shared Drive/Innovation In Software/Citi Training/sales.json' @sf_tut_stage AUTO_COMPRESS=TRUE;
17 PUT file:///C:/Users/kwame/Downloads/sales.json @sf_tut_stage AUTO_COMPRESS=TRUE;
18
19 COPY INTO home_sales(json_column)
20     FROM (SELECT *
```

Results Chart

	JSON_COLUMN
1	{ "location": { "state_city": "MA-Lexington", "zip": "40503" }, "price": "275836", "sale_date": "2017-3-5" }
2	{ "location": { "state_city": "MA-Belmont", "zip": "02478" }, "price": "392567", "sale_date": "2017-3-17" }
3	{ "location": { "state_city": "MA-Winchester", "zip": "01890" }, "price": "389921", "sale_date": "2017-3-21" }

INNOVATION SOFTWARE LEARN. EMPOWER. INNOVATE. Query Details ...

Query duration 359ms

Rows 3

Query ID 01ae8da0-0001-4c0f-0...

44

# POP QUIZ:

## Data Warehouse



Data Warehouses are:

- A: Historical repositories
- B: Transaction processing databases
- C: Query processing engines
- D: A,B, C
- E: A, C



# POP QUIZ:

## Data Warehouse



Data Warehouses are:

- A: Historical repositories
- B: Transaction processing databases
- C: Query processing engines
- D: A,B, C
- E: A, C



# POP QUIZ:

## Data Warehouse



Data warehouses are well-suited for analytics processing:

- A: True
- B: False

# POP QUIZ:

## Data Warehouse



Data warehouses are well-suited for analytics processing:

A: True

B: False



# POP QUIZ:

## Data Warehouse



Which of the following could be a valid Snowflake Account ID:

- A: 98145LBA
- B: LBA6778
- C: LBADSTR
- D: J9871293
- D: None of the above



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Data Warehouse



Which of the following is a key Data Warehouse selection criteria:

- A: Provides data resiliency and recovery
- B: Secures data at rest and in transit
- C: Streamlining the data pipeline
- D: All of the above



# POP QUIZ:

## Data Warehouse



Which of the following is a key Data Warehouse selection criteria:

- A: Provides data resiliency and recovery
- B: Secures data at rest and in transit
- C: Streamlining the data pipeline
- D: All of the above



# POP QUIZ:

## Data Warehouse



How does Snowflake help you to get started:

- A: Snowflake has free courses to get started
- B: Snowflake has a simplified UI designed to help new joiners.
- C: Snowflake has free trial credits that convert to pay-as-you-go.
- D: All of the above



# POP QUIZ:

## Data Warehouse



How does Snowflake help you to get started:

- A: Snowflake has free courses to get started
- B: Snowflake has a simplified UI designed to help new joiners.
- C: Snowflake has free trial credits that convert to pay-as-you-go.
- D: All of the above

# POP QUIZ:

## Data Warehouse



Which of the following is a valid account URL for the BOBSCRAB organization TEST account name in the Ohio (US-EAST-2) AWS region:

- A: test.bobscrab.us-east-2.snowflakecomputing.com
- B: bobscrab.test.ohio-east-2.snowflakecomputing.com
- C: bobscrab.test.azure-us-east-2.snowflakecomputing.com
- D: None of the above

# POP QUIZ:

## Data Warehouse



Which of the following is a valid account URL for the BOBSCRAB organization TEST account name in the Ohio (US-EAST-2) AWS region:

- A: test.bobscrab.us-east-2.snowflakecomputing.com
- B: bobscrab.test.ohio-east-2.snowflakecomputing.com
- C: bobscrab.test.azure-us-east-2.snowflakecomputing.com
- D: None of the above

# Snowflake Access and Setup



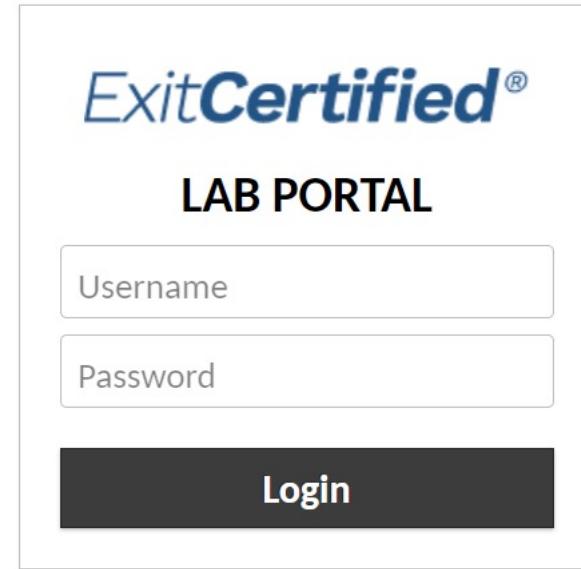
# Access and Setup

- Login to Exit Lab
- Sign-Up for Free Trial for Snowflake
- Log into AWS Account
- Setup Cloud9 IDE
- Clone Git Repository



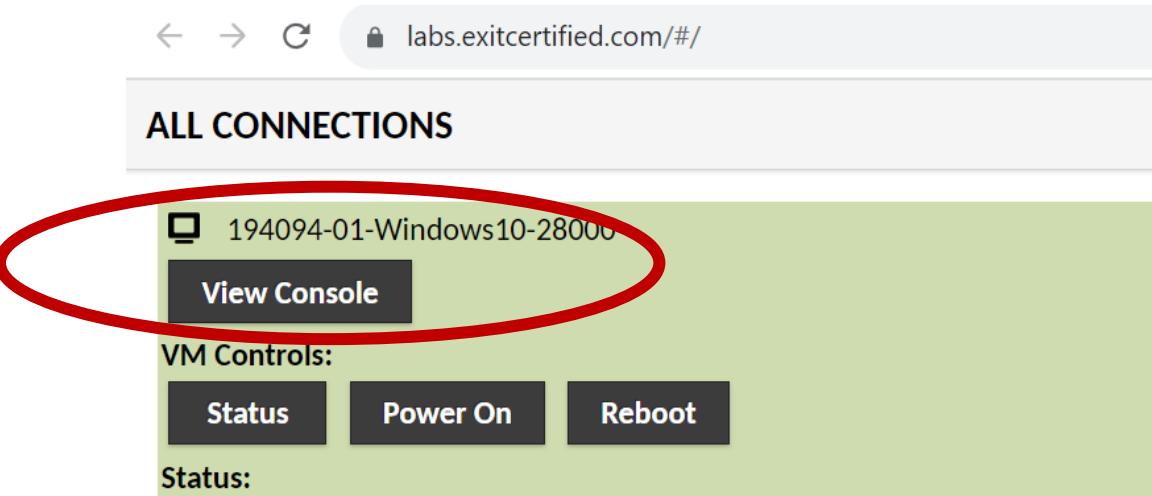
# Login to Exit Lab

- Enter  
<https://labs.exitcertified.com/#/> in your URL
- Sign in with your assigned credentials



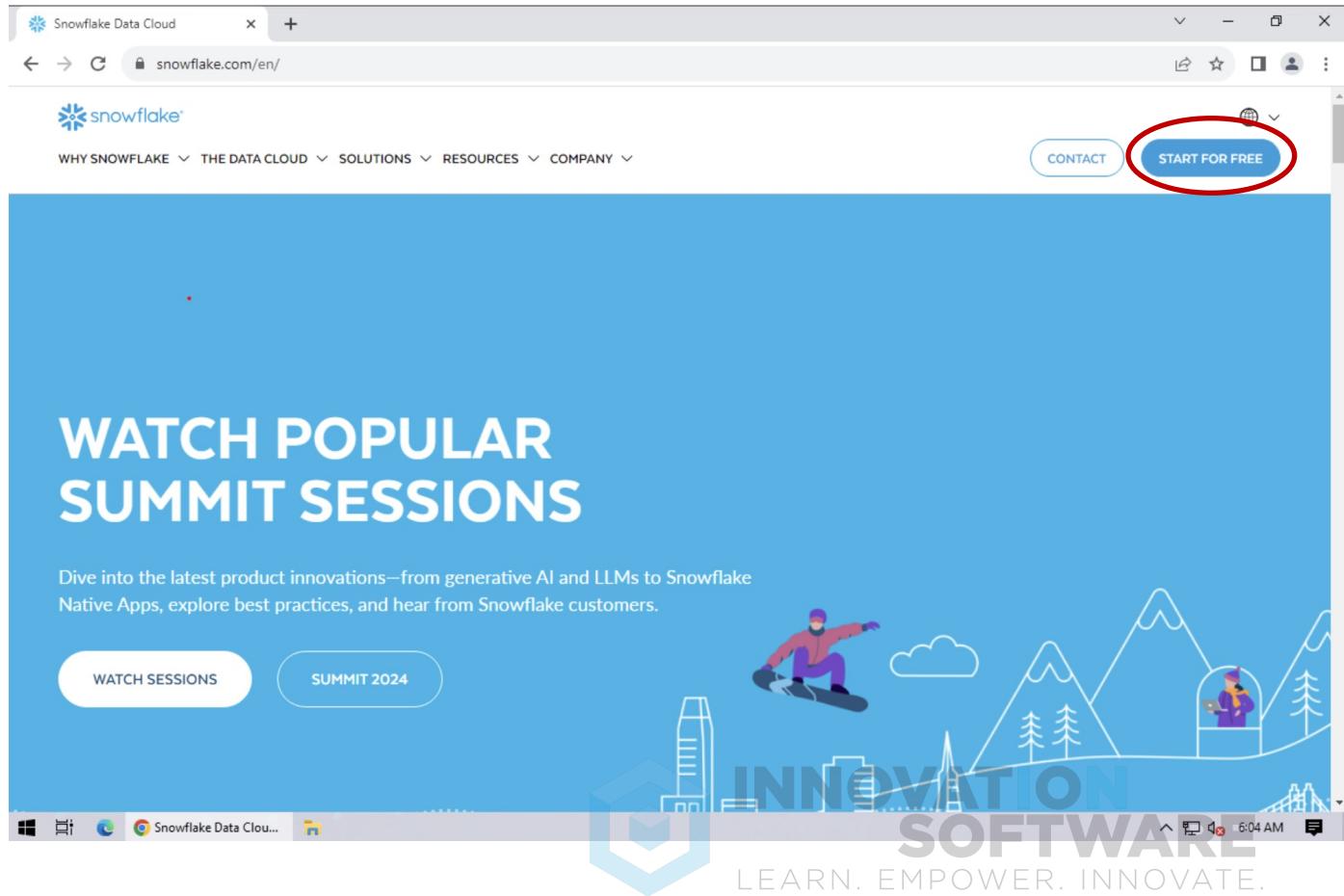
# Login to Exit Lab

- Select “View Console” and it will open you VM for the class



# Sign-Up for Free Trial for Snowflake

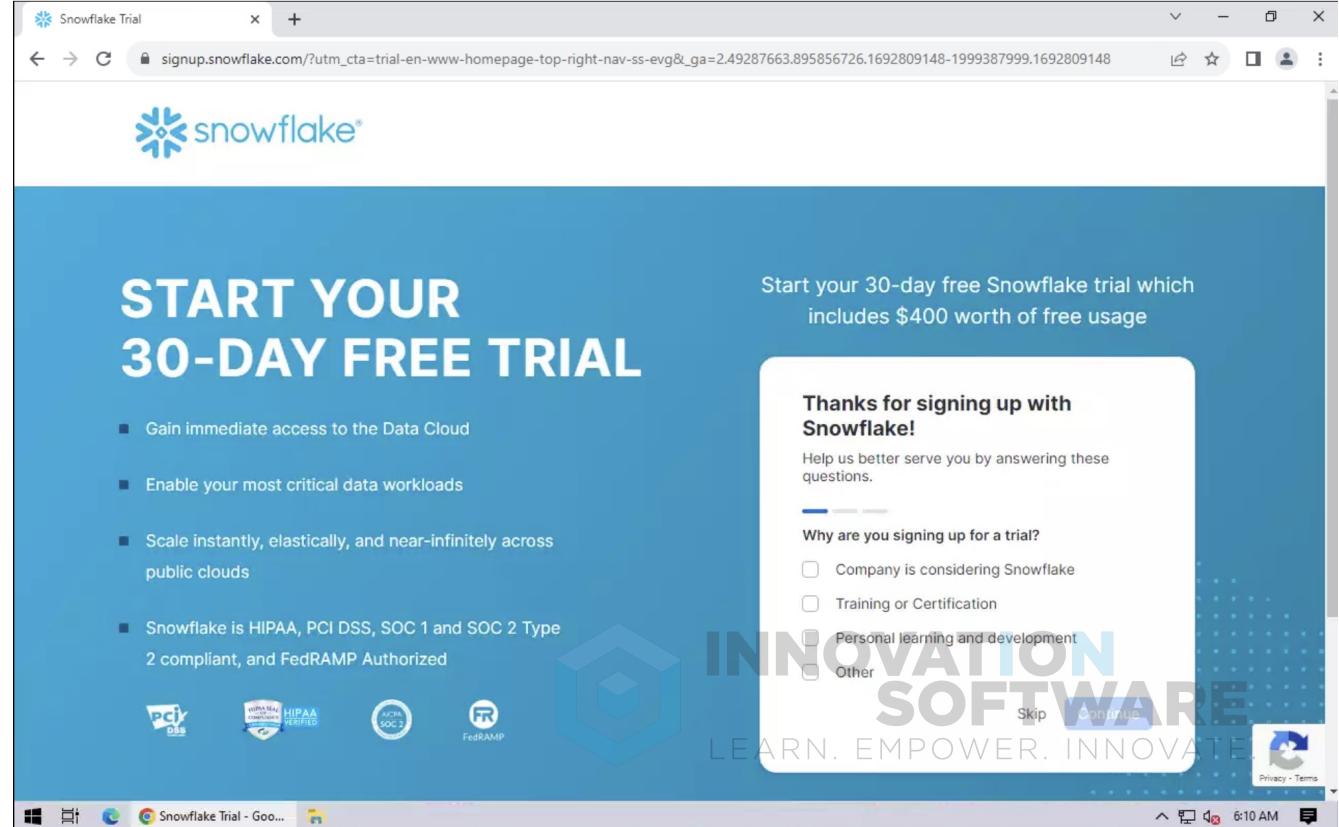
- Go to [www.snowflake.com](http://www.snowflake.com)
- Select “Start for Free” and fill out the form.
- Make sure you select “Business Critical” as your Snowflake edition and “AWS” as the cloud provider



# Sign-Up for Free Trial for Snowflake

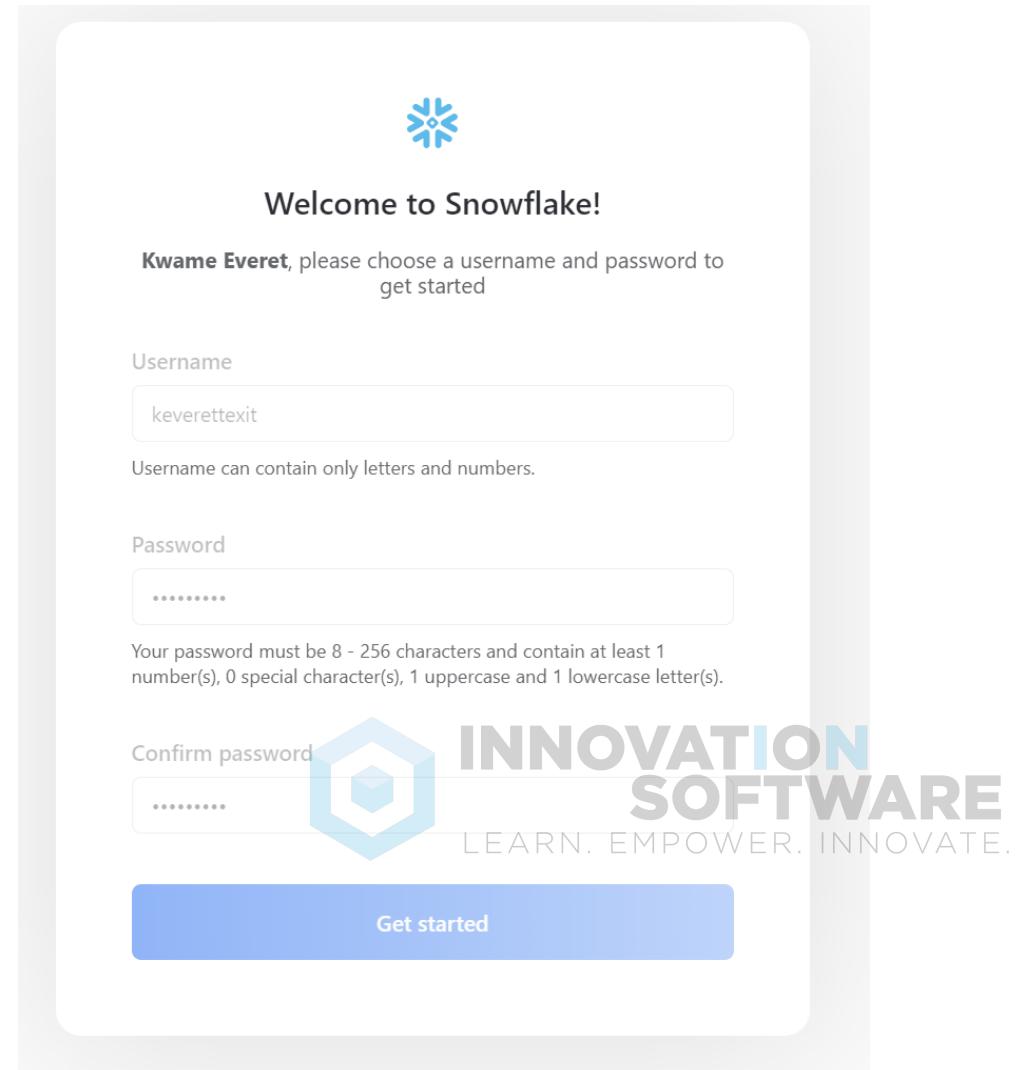
- When done, you'll get an email to activate your account in the next 72 hours. Please check your email to confirm
- You will also receive a unique URL to login like below:

<https://obzvbbvfq20046.snowflakecomputing.com/console/login>



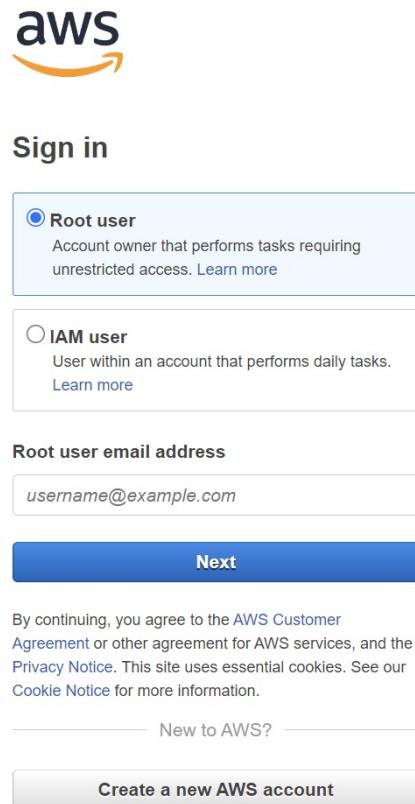
# Sign-Up for Free Trial for Snowflake

- Select “Activate” on the email
- Create user credentials
- Login with the credentials



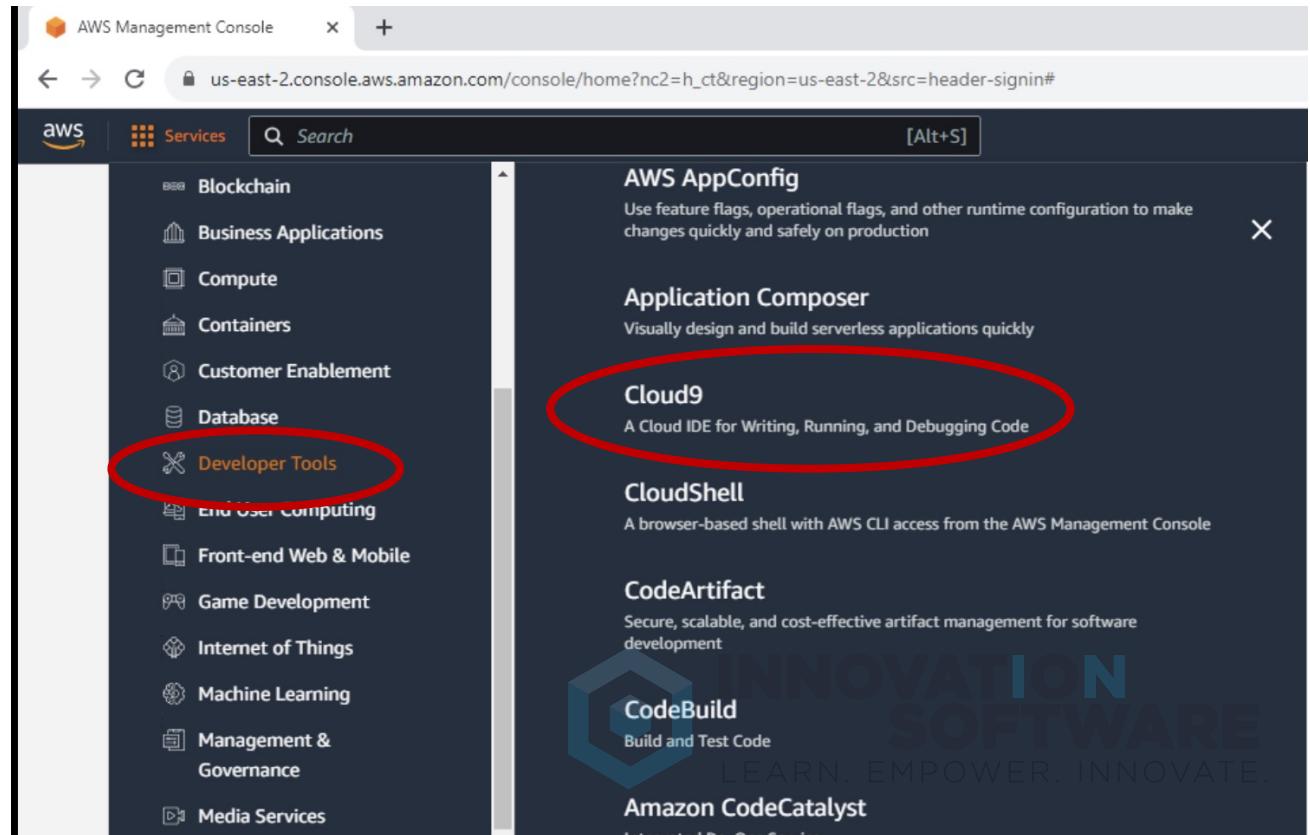
# Log into AWS Account

- Go to:  
[signin.aws.amazon.com](https://signin.aws.amazon.com)
- Login with given course credentials
- Set your region to the region selected for your Snowflake instance



# Setup Cloud9 IDE

- Cloud9 is a cloud-based IDE offered by AWS
- This tool can work with numerous programming languages and connect to a myriad of resources
- To access, search for Cloud9 under AWS services and select



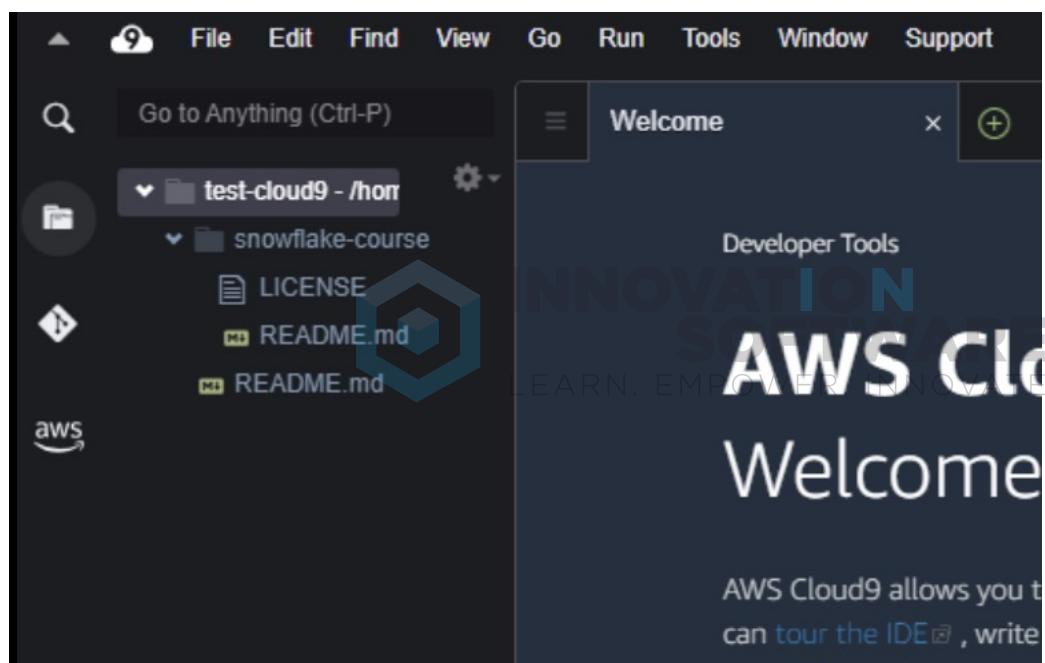
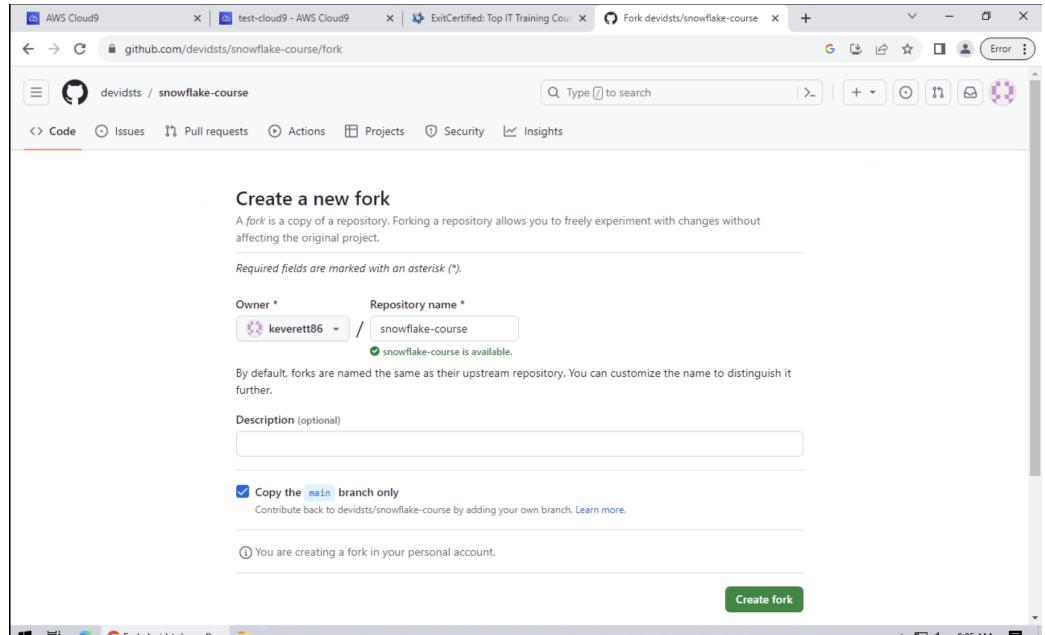
# Setup Cloud9 IDE

- To create a new environment, select “Create New Environment”
- Complete the form:
  - Enter a name
  - Use New EC2 Instance
  - Keep all other default values (we are using Amazon Linux for this course)
  - Select create
  - Open Cloud9 instance after initialized by AWS

The image contains two screenshots of the AWS Cloud9 console. The top screenshot shows the 'Environments' page with a message stating 'No environments' and 'You don't have any environments in us-east-2.'. The bottom screenshot shows the 'Create environment' form. The form has a 'Details' section with fields for 'Name' and 'Description - optional'. Below this is an 'Environment type' section with two options: 'New EC2 instance' (selected) and 'Existing compute'. The 'New EC2 instance' option includes a note: 'Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.' A watermark for 'INNOVATION SOFTWARE LEARN. EMPOWER. INNOVATE.' is visible across the bottom of the screenshots.

# Clone GitHub Repo

- Clone the course GitHub Repo to your personal GitHub account
- Connect your repo to Cloud9



# Architecture and Overview

Kickstart



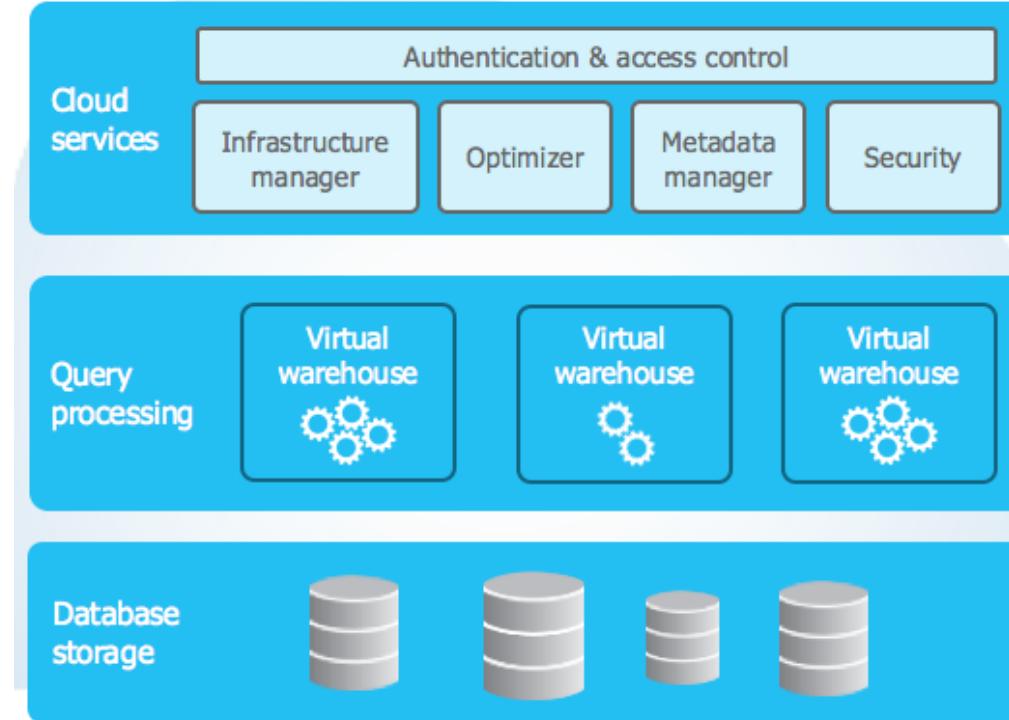
2

# ARCHITECTURE OVERVIEW

- Data Platform as a Cloud Service
- Snowflake Architecture
- Database Storage
- Query Processing
- Cloud Services
- Connecting to Snowflake
- Optimization
- Metadata
- Data Governance

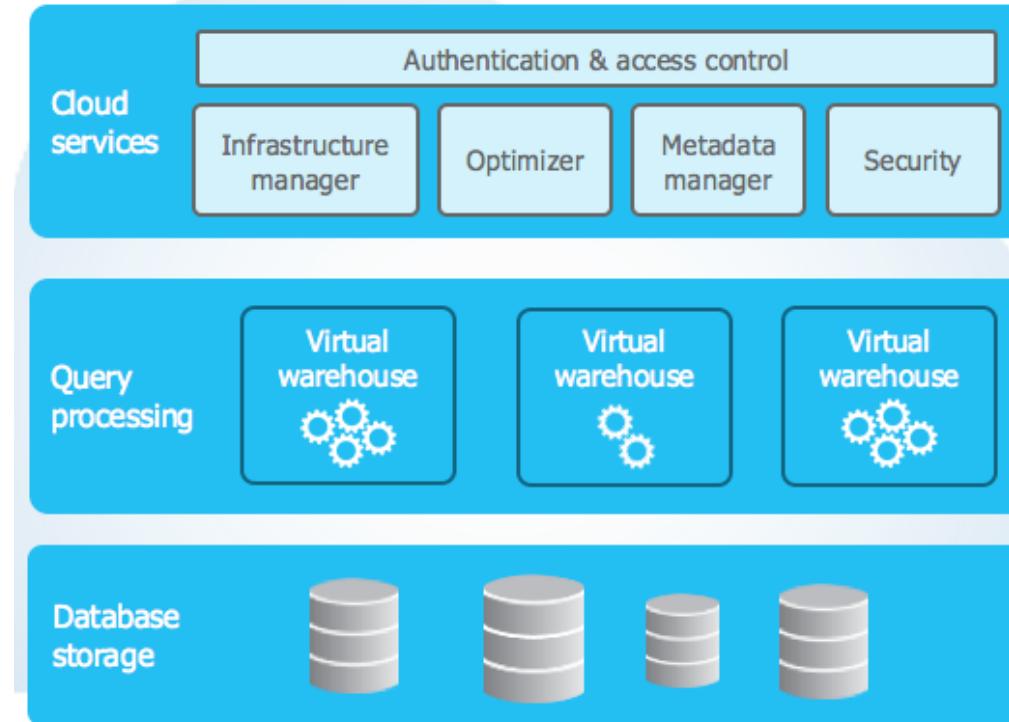
# DATA PLATFORM LAYERS

- Storage Layer - Database Storage
- Computer Layer - Query Processing
- Cloud Services Layer



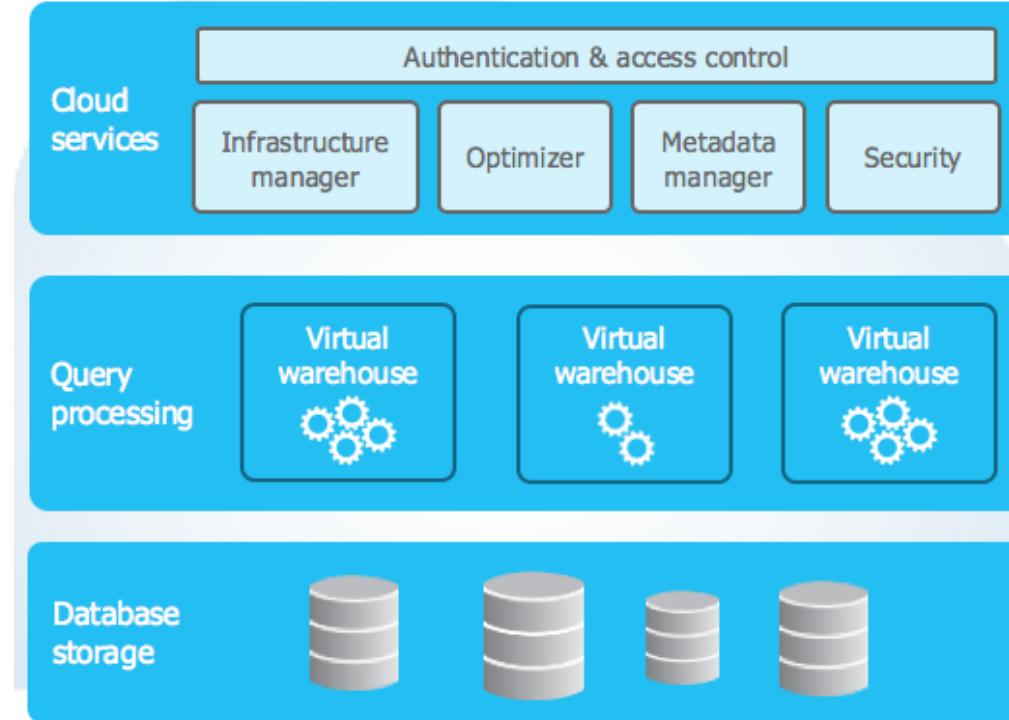
# STORAGE LAYER - DATABASE STORAGE

- Storage Layer - Database Storage
- Computer Layer - Query Processing
- Cloud Services Layer



# DATA PLATFORM LAYERS

- Storage Layer - Database Storage
- Computer Layer - Query Processing
- Cloud Services Layer



# TECHNICAL OVERVIEW

- Understanding Snowflake Table Structures
- Working with Temporary and Transient Tables
- Working with External Tables
- Using the Search Optimization Service
- Overview of Views
- Working with Secure Views
- Working with Materialized Views
- Table Design Considerations
- Cloning Considerations
- Data Storage Considerations

# FRAMING YOUR DATA IN SNOWFLAKE

All data in Snowflake is maintained in databases.

Each database consists of one or more schemas.

Schemas are logical groupings of database objects, such as tables and views.

Snowflake has **no hard limits** on the number of **databases**, **schemas** (within a database), or **objects** (within a schema) you can create.

# DEFAULT DATABASES IN SNOWFLAKE

The following databases are default, although in the trial accounts the SNOWFLAKE DB is not available

- DEMO\_DB
- SNOWFLAKE
- SNOWFLAKE\_SAMPLE\_DATA
- UTIL\_DB

The SNOWFLAKE databases is not available to both SYSADMIN and ACCOUNTADMIN, but rather by default only to the ACCOUNTADMIN user

# SNOWFLAKE TABLE STRUCTURES

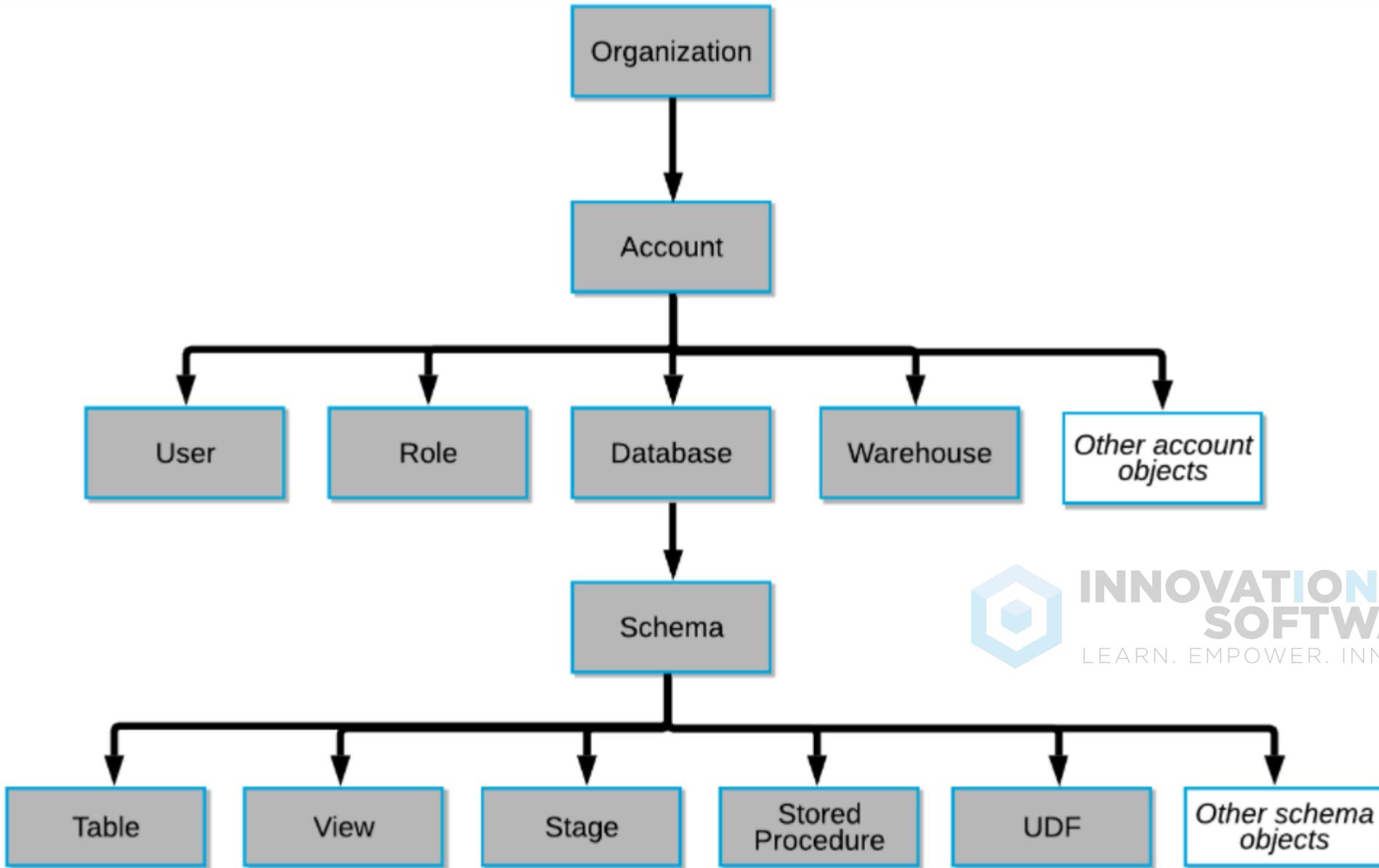
All data in Snowflake is stored in database tables, logically structured as collections of columns and rows.

Snowflake physical table structure features ***micro-partitions*** and ***data clustering***.

These features provide guidance for explicitly defining ***clustering keys*** for very large tables (in the multi-terabyte range) to help optimize table maintenance and query performance.



# SNOWFLAKE OBJECTS



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# MICRO PARTITIONS

Logical Structure

type	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

Physical Structure

Micro-partition 1 (rows 1-6)				Micro-partition 2 (rows 7-12)				Micro-partition 3 (rows 13-18)				Micro-partition 4 (rows 19-24)			
type	2	4	3	type	3	2	4	type	2	4	2	type	1	4	5
name	A	C	C	name	Z	B	C	name	X	Z	Y	name	C	Z	Y
country	UK	SP	DE	country	DE	UK	NL	country	FR	NL	SP	country	FR	NL	SP
date	11/2	11/2	11/2	date	11/2	11/2	11/2	date	11/2	11/2	11/2	date	11/3	11/4	11/4



LEARN. EMPOWER. INNOVATE.

# MICRO PARTITIONS

Logical Structure

type	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

Physical Structure

type	Micro-partition 1 (rows 1-6)			Micro-partition 2 (rows 7-12)			Micro-partition 3 (rows 13-18)			Micro-partition 4 (rows 19-24)		
name	2	4	3	3	2	4	2	4	2	1	4	5
country	A	C	C	Z	B	C	X	Z	Y	C	Z	Y
date	UK	SP	DE	DE	UK	NL	FR	NL	SP	FR	NL	SP
11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/3	11/4	11/4
11/2	11/2	11/2	11/2	11/3	11/3	11/3	11/3	11/3	11/3	11/3	11/5	11/5



LEARN. EMPOWER. INNOVATE.

# MICRO PARTITION BENEFITS

- Snowflake micro-partitions are **derived automatically**
- Micro-partitions are **small in size** (50 to 500 MB, before compression).
- Micro-partitions can **overlap** in their range of **values**.
- Snowflake uses ***columnar storage***.
- Columns are also compressed individually within micro-partitions. Snowflake automatically determines the most efficient compression algorithm for the columns in each micro-partition.



# MICRO PARTITION METADATA

Snowflake stores 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.

# CLUSTER

A clustering key is a subset of columns in a table (or expressions on a table) that are explicitly designated to co-locate the data in the table in the same micro-partitions. This is useful for very large tables where the ordering was not ideal (at the time the data was inserted/loaded) or extensive DML has caused the table's natural clustering to degrade.

Some general indicators to define a clustering key for a table include:

- Queries on the table are running slower than expected or have noticeably degraded over time.
- The clustering depth for the table is large.

A clustering key can be defined at table creation (using the CREATE TABLE command) or afterward (using the ALTER TABLE command). The clustering key for a table can also be altered or dropped at any time.



# CLUSTER BENEFITS

Improved scan efficiency in queries by skipping data that does not match filtering predicates.

Better column compression than in tables with no clustering. This is especially true when other columns are strongly correlated with the columns that comprise the clustering key.

After a key has been defined on a table, no additional administration is required, unless you chose to drop or modify the key. All future maintenance on the rows in the table (to ensure optimal clustering) is performed automatically by Snowflake.

# CLUSTER BENEFITS

Improved scan efficiency in queries by skipping data that does not match filtering predicates.

Better column compression than in tables with no clustering. This is especially true when other columns are strongly correlated with the columns that comprise the clustering key.

After a key has been defined on a table, no additional administration is required, unless you chose to drop or modify the key. All future maintenance on the rows in the table (to ensure optimal clustering) is performed automatically by Snowflake.

# RECLUSTERING

As DML operations (INSERT, UPDATE, DELETE, MERGE, COPY) are performed on a clustered table, the data in the table might become less clustered.

Periodic/regular reclustering of the table is required to maintain optimal clustering.

During reclustering, Snowflake uses the clustering key for a clustered table to reorganize the column data, so that related records are relocated to the same micro-partition. This DML operation deletes the affected records and re-inserts them, grouped according to the clustering key.

**Note – Every time a database table reclusters it will create new micro-partitions, thus generating incremental storage cost. Furthermore, the historical micro-partitions are maintained until continuous data protection features expire**

# RECLUSTERING

As DML operations (INSERT, UPDATE, DELETE, MERGE, COPY) are performed on a clustered table, the data in the table might become less clustered.

Periodic/regular reclustering of the table is required to maintain optimal clustering.

During reclustering, Snowflake uses the clustering key for a clustered table to reorganize the column data, so that related records are relocated to the same micro-partition. This DML operation deletes the affected records and re-inserts them, grouped according to the clustering key.

**Note – Every time a database table reclusters it will create new micro-partitions, thus generating incremental storage cost. Furthermore, the historical micro-partitions are maintained until continuous data protection features expire**

# RECLUSTERING

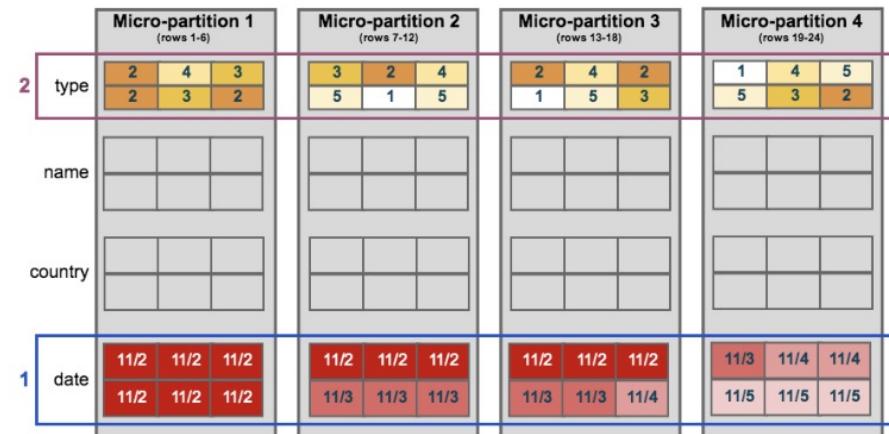
## Logical Structure

type	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

```
SELECT name, country FROM t1  
WHERE type = 2  
AND date = '11/2';
```

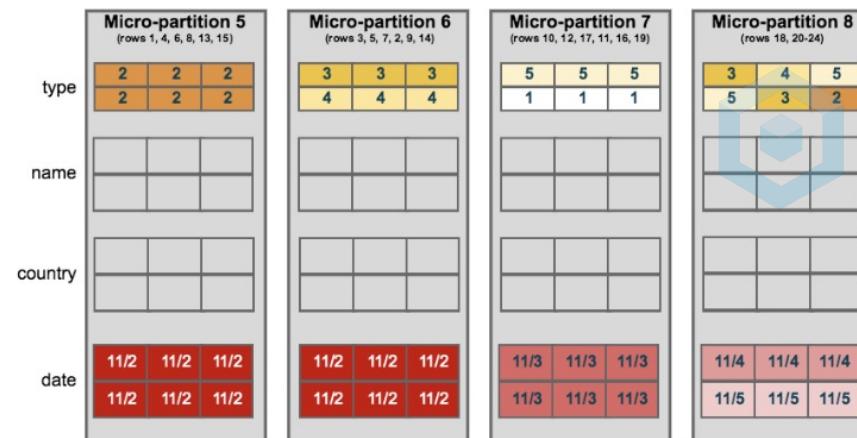
## Physical Structure

## Original Micro-partitions



```
ALTER TABLE t1  
CLUSTER BY (date, type);
```

### New Micro-partitions (After Reclustering)



# INNOVATION SOFTWARE

LEARN. EMPOWER. INNOVATE.

# TEMPORARY AND TRANSIENT TABLES

## Temporary Tables

Snowflake supports creating temporary tables for storing non-permanent, transitory data (e.g. ETL data, session-specific data).

## Transient Tables

Transient tables are specifically designed for transitory data that needs to be maintained beyond each but does not need the same level of data protection and recovery provided by permanent tables.

# TRANSIENT DATABASES AND SCHEMAS

Snowflake also supports creating transient databases and schemas.

All tables created in a transient schema, as well as all schemas created in a transient database, are transient by definition.

# EXTERNAL TABLES

External tables reference data files located in a cloud storage (**Amazon S3, Google Cloud Storage, or Microsoft Azure**) data lake.

**External tables store file-level metadata** about the data files such as the file path, a version identifier, and partitioning information.

This enables querying data stored in files in a data lake as if it were inside a database.



LEARN. EMPOWER. INNOVATE.

# EXTERNAL TABLES USAGES

External tables are **read-only**, therefore no DML operations can be performed on them; however, external tables can be **used for query and join operations**. Views can be created against external tables.

Querying data stored external to the database is likely to be **slower than querying native database tables**



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SEARCH OPTIMIZATION SERVICE

The search optimization service aims to significantly improve the performance of selective point lookup queries on tables. A point lookup query returns only one or a small number of distinct rows.

Use case examples include:

- Business users who need fast response times for critical dashboards with highly selective filters.
- Data scientists who are exploring large data volumes and looking for specific subsets of data.

A user can register one or more tables to the search optimization service. Search optimization is a table-level property and applies to all columns with supported data types (see the list of supported data types further below).

# HOW DOES SNOWFLAKE SEARCH OPTIMIZER SERVICE WORK

To **improve performance for point lookups**, the search optimization service relies on a persistent data structure that serves as an optimized search access path.

A **maintenance service** that runs in the background is responsible for creating and **maintaining the search access path**:

- When you add search optimization to a table, the maintenance service populates the data needed to perform the lookups.
- The process of populating data is done in the background
- When data in the table changes, the service automatically updates the search access path to reflect the changes to the data.
- Queries might run slower if the search access path hasn't updated but will always return up-to-date results.

# QUERY PERFORMANCE OPTIMIZATION

**Clustering:** Can speed any of the following, as long as they are on the clustering key for **Range** and **Equality** searches. A table can be clustered on only a single key.

**Search optimization service:** speeds only **Equality** searches. However, this applies to all the columns of supported types in a table that has search optimization enabled.

**Materialized view:** speeds both **Equality** searches and **Range** searches, as well as some sort operations, but only for the subset of rows and columns included in the materialized view

# QUERY PERFORMANCE COST CONSIDERATIONS

	Storage Cost	Compute Cost
Search Optimization Service	✓	✓
Materialized View	✓	✓
Clustering the Table		

## **VIEW EXAMPLE**

```
create table hospital_table (patient_id integer,  
                            patient_name varchar,  
                            billing_address varchar,  
                            diagnosis varchar,  
                            treatment varchar,  
                            cost number(10,2));  
  
insert into hospital_table  
    (patient_id, patient_name, billing_address, diagnosis, treatment, cost)  
values  
    (1, 'Crabby Bob', 'Fort Meyers Florida', 'Tasty Crustration',  
     'Texas Roadhouse', 2000.00),  
    (2, Harry Potter', 'Gryffindoor', 'Parsel Tongue', 'Horcrux Removal',  
     100000.00)
```

# **VIEW EXAMPLE**

```
create view doctor_view as  
select patient_id, patient_name, diagnosis, treatment from hospital_table;
```

```
create view accountant_view as  
select patient_id, patient_name, billing_address, cost from hospital_table;
```



# SNOWFLAKE VIEWS

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. This is a concept that's replicated from most mature RDBMS products like Oracle and IBM DB2.

Views serve a variety of purposes, including combining, segregating, and protecting data

Snowflake has two types of views:

- Materialized
- Non-materialized



# NON-MATERIALIZED VIEWS

The term “view” generically refers to all types of views; however, the term is used here to refer specifically to non-materialized views.

A view is basically a named definition of a query. A non-materialized view’s results are created by executing a query.

Any query expression that returns a valid result can be used to create a non-materialized view, such as:

- Selecting some (or all) columns in a table.
- Selecting a specific range of data in table columns.
- Joining data from two or more tables.

# SECURE VIEWS

Snowflake optimizations for views require access to the underlying data in the base tables for the view.

Secure views do not utilize these optimizations, ensuring that users have no access to the underlying data.

For security or privacy reasons, you might not wish to expose the underlying tables or internal structural details for a view.

With secure views, the view definition and details are only visible to authorized users

# WHEN TO USE SECURE VIEWS

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, such as views created for simplifying querying data for which users do not need to understand the underlying data representation. This is because the Snowflake query optimizer, when evaluating secure views, bypasses certain optimizations used for regular views. This might result in some impact on query performance for secure views.

# TABLE DESIGN CONSIDERATIONS

- Date/Time Data Types for Columns
- Referential Integrity Constraints
- When to Set a Clustering Key
- When to Specify Column Lengths
- Storing Semi-structured Data in a VARIANT Column vs. Flattening the Nested Structure
- Converting a Permanent Table to a Transient Table or Vice-Versa

# CLONING CONSIDERATIONS

A cloned object does not retain any granted privileges on the source object itself (i.e. clones do not automatically have the same privileges as their sources).

A system administrator or the owner of the cloned object must explicitly grant any required privileges to the newly-created clone.

However, if the source object is a database or schema, for child objects contained in the source, the clone replicates all granted privileges on the corresponding child objects:

- For databases, contained objects include schemas, tables, views, etc.
- For schemas, contained objects include tables, views, etc.

# DATA STORAGE – CONTINUOUS DATA PROTECTION (CDP)

CDP, which includes Time Travel and Fail-safe, is a standard set of features available to all Snowflake accounts at no additional cost.

However, because your account is charged for all data stored in tables, schemas, and databases created in the account, CDP does have an impact on storage costs, based on the total amount of data stored and the length of time the data is stored.

Storage is calculated and charged for data regardless of whether it is in the Active, Time Travel, or Fail-safe state. Because these life-cycle states are sequential, updated/deleted data protected by CDP will continue to incur storage costs until the data leaves the Fail-safe state.

# MANAGING COSTS – SHORT LIVED DATA

**CDP is designed to provide long-term protection for your data.** This data is typically stored in permanent tables. Unless otherwise specified at the time of their creation, tables in Snowflake are created as permanent.

During an ETL or data modeling process, tables may be created that are short-lived. For these tables, it does not make sense to incur the storage costs of CDP. **Snowflake provides two separate mechanisms to support short-lived tables:**

- Temporary tables
- Transient tables

# MANAGING COSTS – RESILIENT USEFUL DATA

**Fact or Dimension** tables, which have **different usage patterns** and, therefore, different storage considerations:

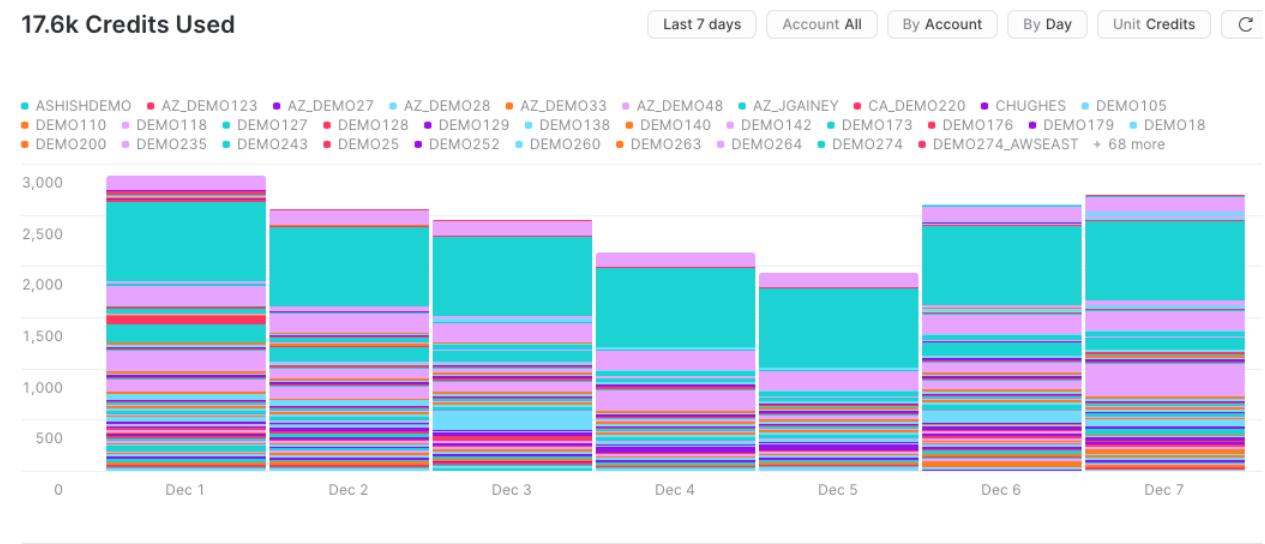
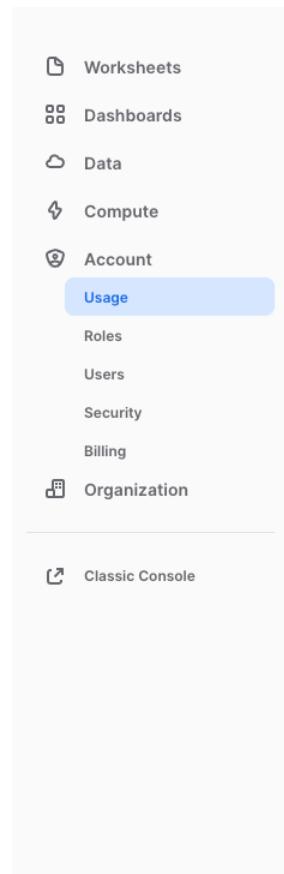
**Fact tables** are typically very large in size and experience a low degree of churn (row updates or deletes). Most changes to fact tables are inserts of new data or, in some cases, deletions of older data. CDP is ideal for fact tables as it provides full data protection at a very low storage cost.

**Dimension tables** have a different update pattern. Row updates and deletions are much more common in dimension tables.



# OPTIMIZATION

You can explore historical performance using Snowsight or by writing queries against views in the ACCOUNT\_USAGE schema. A user without access to the ACCOUNT\_USAGE schema can query similar data using the Information Schema.



Credit Consumption per Account from 12/1/2021 to 12/7/2021



# SNOWSIGHT

Snowflake allows the ability to create dashboards using Snowsight

Snowsight lets to query different tables and create simple dashboards

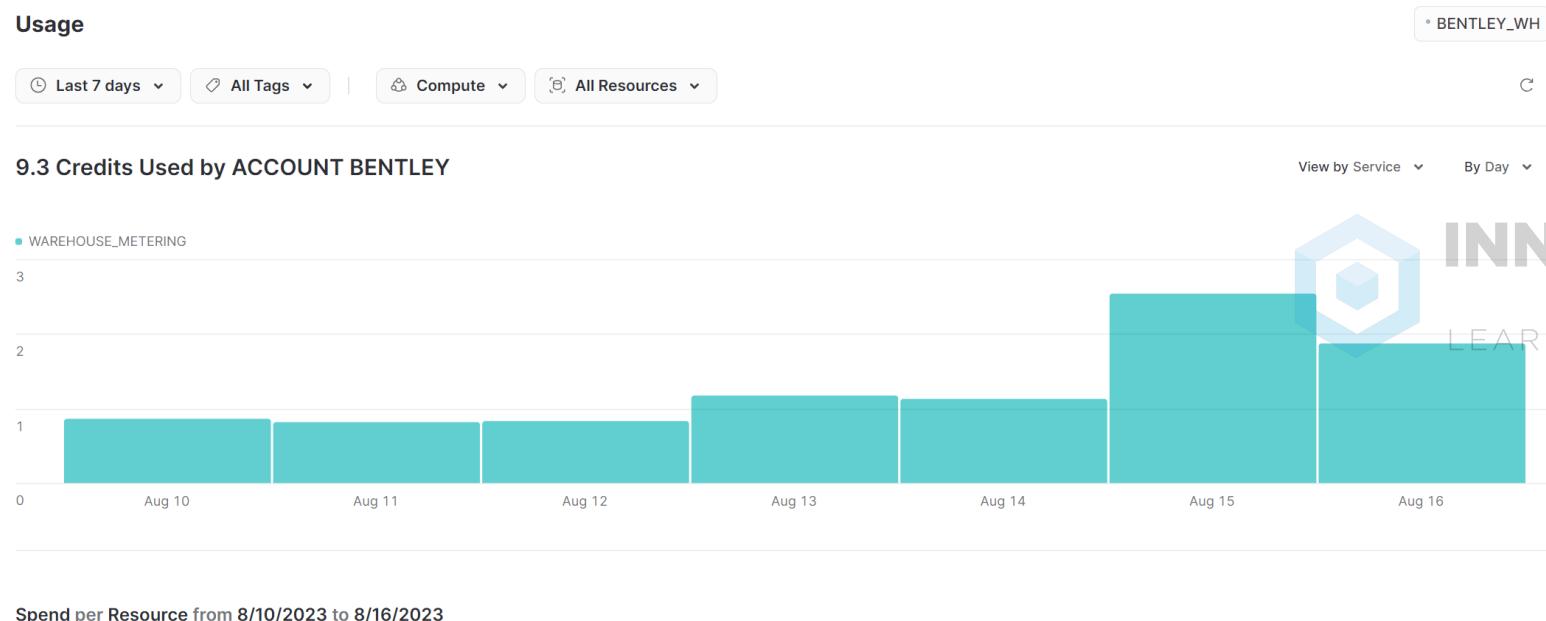
Snowsight combined with Account Usage is a great method of creating monitoring dashboards in Snowflake

The screenshot shows the Snowsight interface. On the left, a sidebar menu includes 'Worksheets', 'Dashboards' (which is selected and highlighted in blue), 'Apps', 'Data', 'Marketplace', 'Activity', 'Admin', and 'Help & Support'. A promotional banner for a 30-day trial is visible. The main area is titled 'Dashboards' and shows a list of recent dashboards. One dashboard titled 'test-snowsight' is listed, showing it was viewed just now by ACCOUNTADMIN. Below this, there are two cards: a table card and a chart card. The table card, titled '2023-08-24 5:40am', displays data from a table with columns CC\_CALL\_CENTER\_SK, CC\_CALL\_CENTER\_ID, CC\_REC\_START\_DATE, and CC\_REC\_END\_DATE. The chart card, also titled '2023-08-24 5:40am', is a bar chart showing values over time, with a watermark reading 'INNOVATION SOFTWARE LEARN. EMPOWER. INNOVATE.'.

# ACCOUNT USAGE QUERY

The [Account Usage](#) schema contains views related to the execution times of queries and tasks. It also contains a view related to the load of a warehouse as it executes queries. You can write queries against these views to drill down into performance data and create custom reports and dashboards.

By default, only the account administrator (i.e. user with the ACCOUNTADMIN role) can access views in the ACCOUNT\_USAGE schema. To allow other users to access these views, refer to [Enabling Snowflake Database Usage for Other Roles](#).



# ACCOUNT USAGE QUERY

ACCOUNT_USAGE View	Description	Latency
QUERY_HISTORY	Used to analyze the Snowflake query history by various dimensions (time range, execution time, session, user, warehouse, etc.) within the last 365 days (1 year).	Up to 45 minutes
WAREHOUSE_LOAD_HISTORY	Used to analyze the workload on a warehouse within a specified date range.	Up to 3 hours
TASK_HISTORY	Used to retrieve the history of task usage within the last 365 days (1 year).	Up to 45 minutes



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# ACCOUNT USAGE QUERY - EXAMPLE

```
1  SELECT query_id,
2      ROW_NUMBER() OVER(ORDER BY partitions_scanned DESC) AS query_id_int,
3      query_text,
4      total_elapsed_time/1000 AS query_execution_time_seconds,
5      partitions_scanned,
6      partitions_total
7  FROM snowflake.account_usage.query_history Q
8  WHERE TO_DATE(Q.start_time) > DATEADD(day,-1,TO_DATE(CURRENT_TIMESTAMP()))
9      AND total_elapsed_time > 0 --only get queries that actually used compute
10     AND error_code IS NULL
11     AND partitions_scanned IS NOT NULL
12 ORDER BY total_elapsed_time desc
13 LIMIT 50;
```

↳ Results    ↵ Chart    🔍 ⏪ ⏴ ⏵ ⏷

	QUERY_ID	...	QUERY_ID_INT	QUERY_TEXT	
1	01ae5d20-0000-c731-0000-035d01e89006		6	insert into T0701T800MD000 select a12.ID_TME_FIS_YYYYWW ID_TME_FIS_YYY	
2	01ae5d20-0000-c734-0000-035d01e8800a		1	select a12.ID_TME_FIS_YYYY ID_TME_FIS_YYYY, a12.ID_TME_FIS_YYYYMM ID_TM	
3	01ae5d98-0000-c73f-0000-035d01e8b09a		5	insert into TEV01IY2WMD000 select a12.ID_TME_FIS_YYYYWW ID_TME_FIS_YYYY	
4	01ae5d25-0000-c732-0000-035d01e8a096		3	insert into TZB01K3NSMD001 select a12.ID_TME_FIS_YYYYWW ID_TME_FIS_YYY	
5	01ae5d9b-0000-c731-0000-035d01e89166		2	insert into TK701IN88MD001 select a12.ID_TME_FIS_YYYYWW ID_TME_FIS_YYYY	
6	01ae5db6-0000-c73f-0000-035d01e8b112		41	insert into TRB01YLFAMD001 select a13.DS_GOA_SEARCH_KEYWORD DS_GOA_S	
7	01ae5dd5-0000-c734-0000-035d01e882a2		26	insert into T0N014YO0SP00Q select a11.DATE_STR_TRAFFIC DT_TME_YYYYMMDD	
8	01ae5dh6-0000-c71c-0000-035d01e8715e		63	insert into "MSTR DM DAI Y SAI FS VARTO PI AN" select "a11" "ID PRD DEPARTN	

INNOVATION  
SOFTWARE

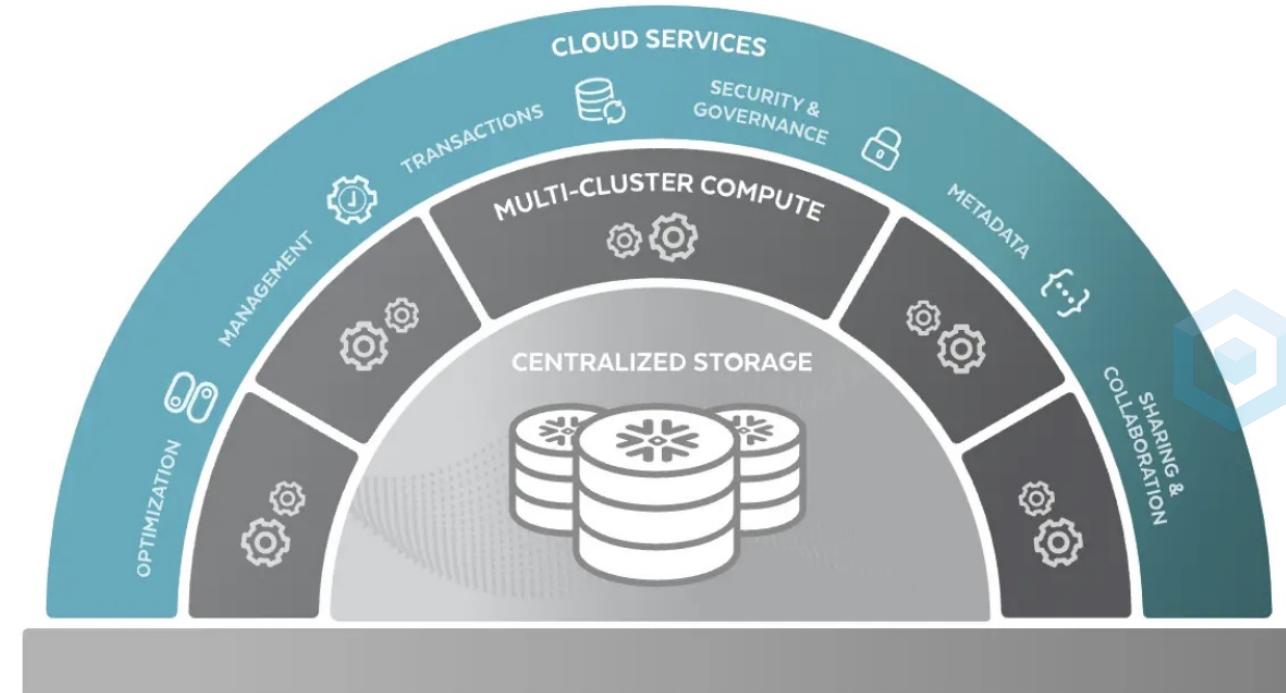
LEARN. EMPOWER. INNOVATE.

QUERY\_ID A  
100% filled

QUERY\_ID INT #

# METADATA

Snowflake metadata management is a part of the data governance discipline which involves processes, policies, workflows, and technology to identify, organize, and surface Snowflake metadata to data consumers. Metadata management is the key to adding actionable context to the assets in your [Snowflake data warehouse](#).



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# METADATA

The INFORMATION SCHEMA in each database provides baseline metadata for all the objects in Snowflake. These can be queried directly.

The screenshot shows the Snowflake UI interface. On the left, the sidebar displays 'Databases' and 'Worksheets'. Under 'SNOWFLAKE\_SAMPLE\_DATA', the 'INFORMATION\_SCHEMA' schema is expanded, and 'COLUMNS' is selected. The main area shows a query editor with the following SQL code:

```
1 SELECT *
2 FROM SNOWFLAKE_SAMPLE_DATA.INFORMATION_SCHEMA.COLUMNS
```

The 'Results' tab is selected, displaying the output of the query:

COLUMNS	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	CC
A TABLE_CATALOG	VARCHAR(16777216)					
↑ ⓘ Database that the table belongs to						
A TABLE_SCHEMA	VARCHAR(16777216)					
↑ ⓘ Schema that the table belongs to						
A TABLE_NAME	VARCHAR(16777216)					
↑ ⓘ Table that the column belongs to						
A COLUMN_NAME	VARCHAR(16777216)					
↑ ⓘ Column name						

On the right, there is a 'Query Details' panel with the following information:

- Query duration: 1.6s
- Rows: 1.5K
- Query ID: 01ae5ff9-0000-c72d-0...
- TABLE\_CATALOG: SNOWFLAKE\_SAMPLE\_DATA
- TABLE\_SCHEMA: TPCDS\_SF100TCL
- TABLE\_NAME: CUSTOMER
- COLUMN\_NAME: C\_EMAIL\_ADDRESS
- ORDINAL\_POSITION: 17
- CC: nul



INNOVATION  
SOFTWARE

LEARN. EMPOWER. INNOVATE.

# DATA CATALOG

A data catalog is an organized inventory of data assets in the organization. It uses metadata to help organizations manage their data. It also helps data professionals collect, organize, access, and enrich metadata to support data discovery and governance. Snowflake doesn't have an internal data catalog tool.

When you share data via Snowflake Marketplace, the data is automatically cataloged for external users to leverage. However, there are many data catalog tools to leverage with Snowflake.



# DATA CATALOG



Atlan is the first data catalog to be validated as a Snowflake Ready Technology Partner. Atlan enables you to manage your Snowflake data governance with numerous features including:

- Ability to create custom classifications tags, including PII, HIPPA, GDPR and other standards
- Auto-identify and classify sensitive data in your data warehouse
- Ability to set masking policies
- Autocatalog and organize the data warehouse so that consumers can understand the data they want to access

The screenshot shows the Atlan Data Catalog interface. On the left, under 'Personas', there are five entries: Data Consultant, Data Engineer, Finance Team, Marketing Team, and Data Analyst. In the center, under 'Data Consultant', there is an 'Overview' section with 'Policies 20', 'Metadata 13', 'Data 7', and four sample policies (Sample policy 1, Sample policy 2, Sample policy 3, Sample policy 4). On the right, a modal window titled 'Manage permissions' is open, showing options for 'Assets' (Read, Update), 'Governance' (Update Classifications, Add Terms, Remove Terms), and descriptions for each.

**INNOVATION SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

- 
- |  |  |  |
|--|--|--|
| <br><b>Metadata policy</b><br>Control who can read and edit metadata about your Snowflake assets. | <br><b>Data policy</b><br>Define who can view and query your Snowflake data, with masking policies. | <br><b>Glossary policy</b><br>Manage who can edit data definitions, metric formulas, and business taxonomies. |
|--|--|--|

# DATA CATALOG



Dataedo is a solution to create a fully integrated and standardize data catalog across numerous data warehouses. The tool can catalog data not only from Snowflake, but all data sources across the organization creating a single source of truth. This creates a holistic view and enable ER-diagrams, cataloging and management of data across all sources

A screenshot of the Dataedo Catalog interface. The left sidebar shows a list of data sources including Adventure Works Tabular, AdventureWorks Azure SQL, AdventureWorks Database, Bureau of Labor Statistics Google Sheets, Company Data Lake, company-data-lake AmazonS3, CompanyDB IBM Db2, ContosoDW Azure Synapse, Crawled S3 AWS Athena, Cycling Cassandra, Dataedo Dataverse, Dataedo Dynamics365, Dataedo Salesforce, dbt, dbt Warehouse (Google BigQuery), DVD Rental Shop MariaDB, and DVD Rental Shop MySQL. The main area displays an Entity-Relationship (ER) diagram for the "HumanResources.Employee" table. The diagram shows various tables like "HumanResources.Employee", "Person.Person", "Sales.SalesPerson", "HumanResources.JobCandidate", "Production.Document", and "Events AWS Redshift" connected by relationships. A tooltip for the "BusinessEntityID" column indicates it is a primary key with a unique identifier type and a not null constraint. The interface has tabs for Overview, Columns, Diagram, Data lineage, Referencing (FKs), Referenced, Keys, and T.

A screenshot of the Dataedo Column details interface for the "rowguid" column. The left sidebar shows a list of data sources including AdventureWorks Azure SQL, Address, and the current column "rowguid". The main area displays detailed information about the "rowguid" column, including its Type (uniqueidentifier, Not null), Data Insights (Table rows count: 19,614, Distribution: 100% UNIQUE), Description (ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample.), and Keys. A large watermark in the center reads "INNOVATION SOFTWARE LEARN. EMPOWER. INNOVATE.". On the right, there is a "Community" section with a comment from Richard: "We need a user-friendly definition here" and a reply from John Doe: "Thanks for pointing that out. I just provided one!". There are also "Report Catalog", "Subject Areas", "Reference Data", "Classification", and "Community" sections on the left.

# DATA CATALOG



Collibra is an industry leading platform for data governance and catalog across numerous sources, including Snowflake. Collibra enables many core governance tasks including data discovery, governance, lineage and quality management. Collibra has Snowflake industry certification with financial services, with numerous use cases in the industry



**INNOVATION SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

The screenshot shows the Collibra Data Catalog interface. At the top, there's a navigation bar with icons for Home, Catalog, Create, Search, and other functions. A user profile for Joanna Zhou is visible on the right. The main area displays a data set titled "Credit Risk Rating and Customer Data". The data set is categorized as a "Data Set" with a status of "under review". It includes a "Log data issue" button and a "Add to data basket" link. Below the title, there are filters for "Period" (Month), "Alert type" (All), and "Source" (Custom & automated). A table lists various alerts: GDPR, Missing values, Out of range, and Negative values. Each alert includes details like the column, date, alert type, threshold, value, level, and a "log issue" link. On the left, there's a sidebar with icons for Home, Alert, Log, Overview, Sample violations, Pattern mismatch, Unique values, Skewness, and Type mismatch. A large green bar chart at the bottom is labeled "Overview 02/21" and shows "Total" and "Violation" counts over time from 02/15 to 02/21. The chart has a legend for "History" (green) and "Total" (green) and "Violation" (red).

# DATA GOVERNANCE

Data governance is an organization's management of its data availability, usability, consistency, and data integrity and data security. It defines who can take what action, upon what data, in what situations, using what methods and includes the processes, roles, policies, standards, and metrics for ensuring effective data management throughout the lifecycle of the data and for its use by the entire organization. Effective data governance empowers users to develop business insights from high-quality, secure



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# DATA GOVERNANCE - ROLES

Data governance requires teamwork with deliverables from all your departments. Clearly defined roles are essential to every data governance program, and it is important to assign levels of ownership across your organization. Common roles might include:

- **Data governance council (steering committee/strategic level):** Strategic guidance of the data governance program
- **Data governance board (tactical level):** Develops an organization's policies and practices to treat data as a strategic asset.
- **Data managers:** Creates database systems that meet an organization's needs
- **Data owners:** Individual who is accountable for a data asset.
- **Data stewards:** Responsible for utilizing your data governance processes to ensure the quality of data elements, including content and metadata.
- **Data users:** Data users are team members with direct responsibility for entering and using data as part of their daily tasks.

# DATA GOVERNANCE

Snowflake has many native tools to promote data governance on the platform. We are going to highlight a few key areas below:

- **Data Sensitivity & Access Visibility**

- Object Tagging
- Data Classification

- **Data Access Policies**

- Masking Policies
- Row Access Policies

- **Data Lineage & Dependencies**

- Access History
- Object Dependencies

# DATA GOVERNANCE - OBJECT TAGGING

Tags enable data stewards to monitor sensitive data for compliance, discovery, protection, and resource usage use cases through either a centralized or decentralized data governance management approach.

A tag is a schema-level object that can be assigned to another Snowflake object. A tag can be assigned an arbitrary string value upon assigning the tag to a Snowflake object. Snowflake stores the tag and its string value as a key-value pair. The tag must be unique for your schema, and the tag value is always a string.



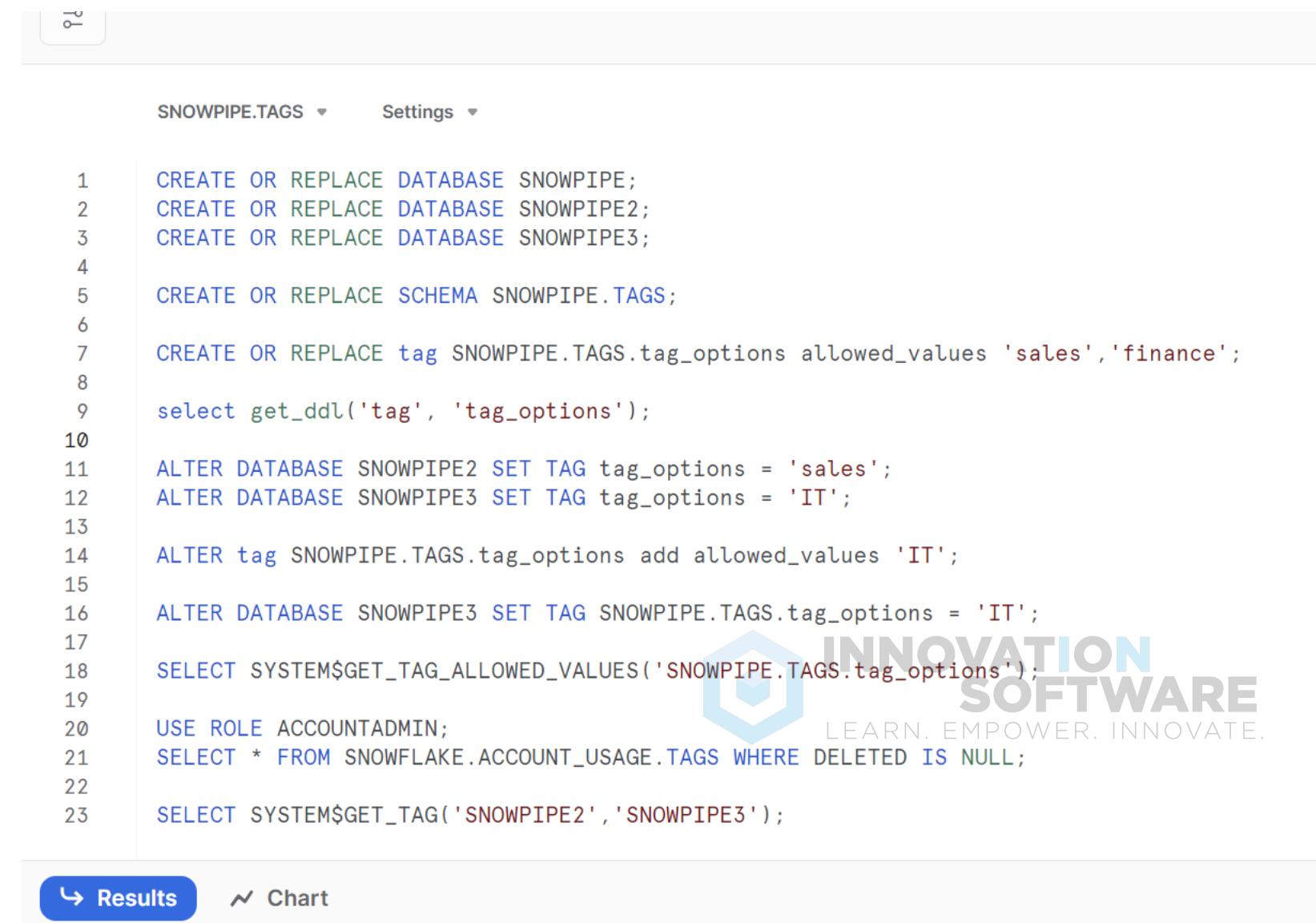
# DATA GOVERNANCE - OBJECT TAGGING

Example – Creating Tags on Sample Databases

We can use the CREATE TAG to generate tags and set particular value options

We can only apply approve values on the objects

Tags are visible in the Account\_Usage table



The screenshot shows a Snowflake interface with a code editor. The title bar says "SNOWPIPE.TAGS". The code editor contains the following SQL script:

```
1 CREATE OR REPLACE DATABASE SNOWPIPE;
2 CREATE OR REPLACE DATABASE SNOWPIPE2;
3 CREATE OR REPLACE DATABASE SNOWPIPE3;
4
5 CREATE OR REPLACE SCHEMA SNOWPIPE.TAGS;
6
7 CREATE OR REPLACE tag SNOWPIPE.TAGS.tag_options allowed_values 'sales','finance';
8
9 select get_ddl('tag', 'tag_options');
10
11 ALTER DATABASE SNOWPIPE2 SET TAG tag_options = 'sales';
12 ALTER DATABASE SNOWPIPE3 SET TAG tag_options = 'IT';
13
14 ALTER tag SNOWPIPE.TAGS.tag_options add allowed_values 'IT';
15
16 ALTER DATABASE SNOWPIPE3 SET TAG SNOWPIPE.TAGS.tag_options = 'IT';
17
18 SELECT SYSTEM$GET_TAG_ALLOWED_VALUES('SNOWPIPE.TAGS.tag_options');
19
20 USE ROLE ACCOUNTADMIN;
21 SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.TAGS WHERE DELETED IS NULL;
22
23 SELECT SYSTEM$GET_TAG('SNOWPIPE2', 'SNOWPIPE3');
```

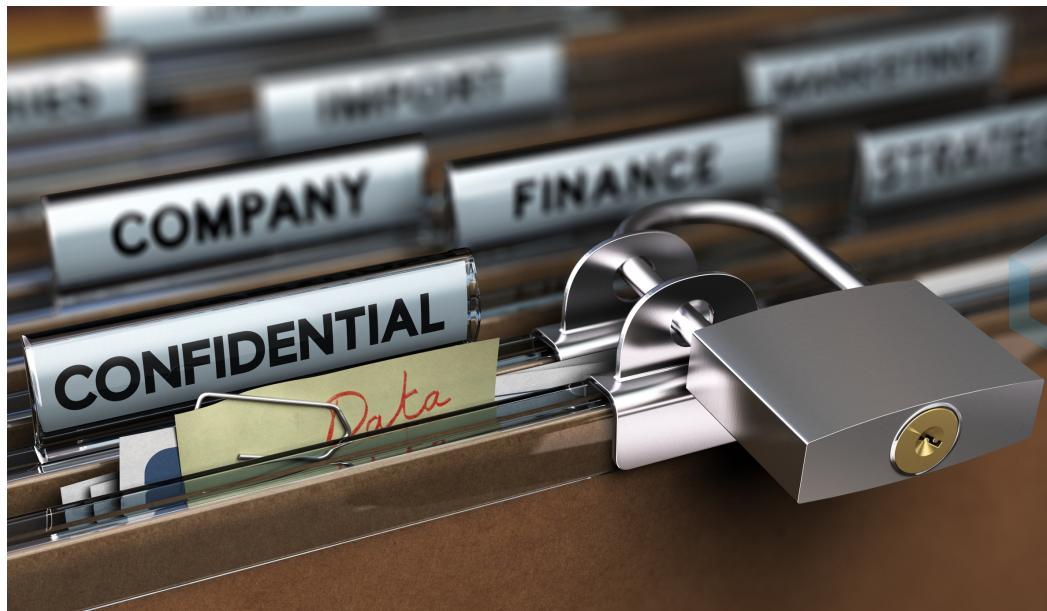
At the bottom of the interface, there are two buttons: "Results" and "Chart".



# DATA GOVERNANCE - CLASSIFICATION

Classification is a multi-step process that associates Snowflake-defined tags (i.e. system tags) to columns by analyzing the cells and metadata for personal data; this data can now be tracked by a data engineer.

Based on the tracking information and related audit processes, the data engineer can protect the column containing personal or sensitive data with a masking policy or the table containing this column with a row access policy.

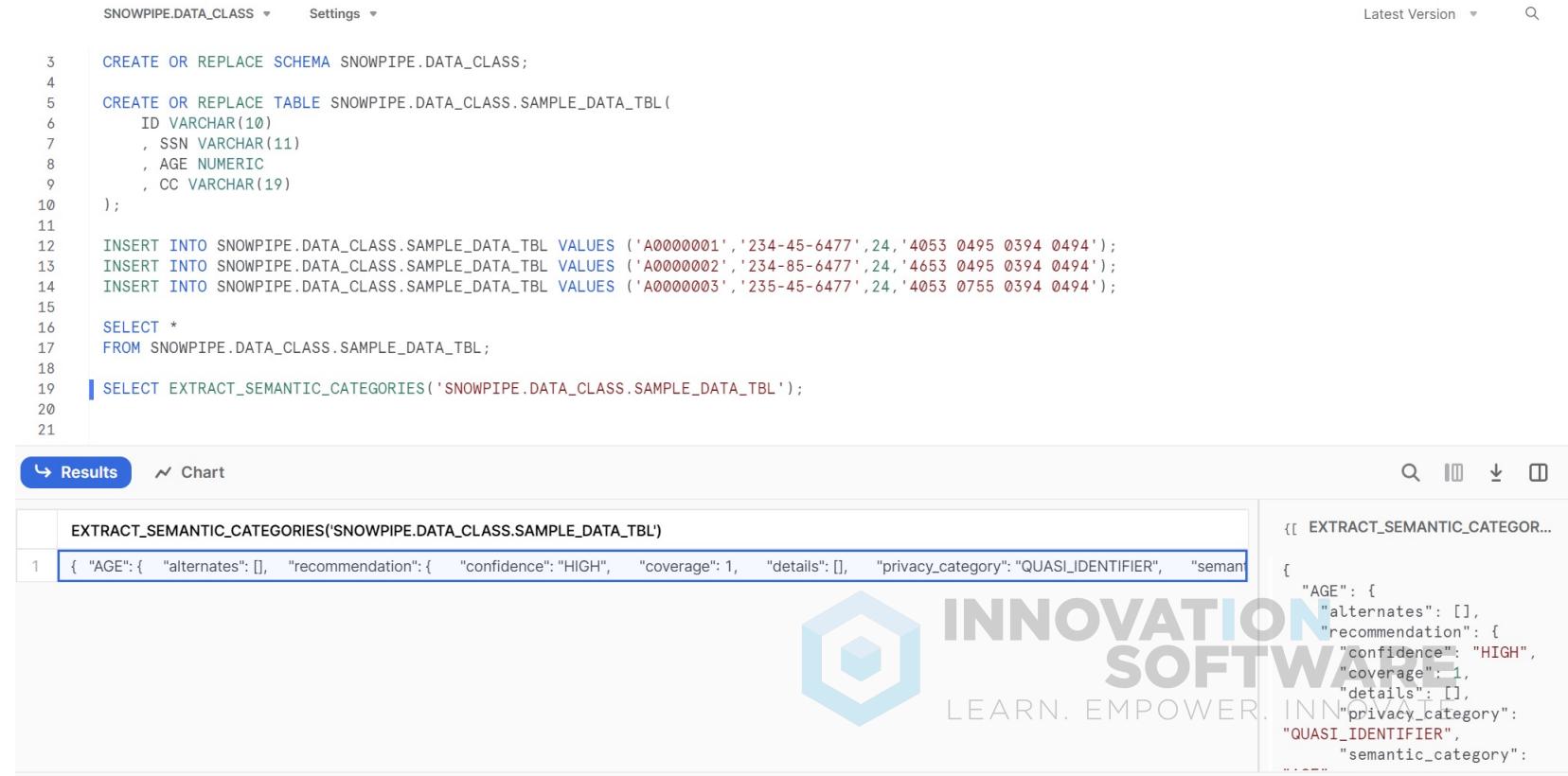


# DATA GOVERNANCE - CLASSIFICATION

We can create a dummy table with SSN, AGE, Credit Card and other data

Extract\_SEMANTIC\_CATEGORIES will scan the table and attempt to classify each column accordingly

The results are in a JSON



Snowflake UI screenshot showing the creation of a schema and table, and the execution of the EXTRACT\_SEMANTIC\_CATEGORIES function.

Code:

```
SNOWPIPE.DATA_CLASS Settings ▾
Latest Version ▾ Q
CREATE OR REPLACE SCHEMA SNOWPIPE.DATA_CLASS;
CREATE OR REPLACE TABLE SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL(
    ID VARCHAR(10)
    , SSN VARCHAR(11)
    , AGE NUMERIC
    , CC VARCHAR(19)
);
INSERT INTO SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL VALUES ('A0000001','234-45-6477',24,'4053 0495 0394 0494');
INSERT INTO SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL VALUES ('A0000002','234-85-6477',24,'4653 0495 0394 0494');
INSERT INTO SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL VALUES ('A0000003','235-45-6477',24,'4053 0755 0394 0494');
SELECT *
FROM SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL;
SELECT EXTRACT_SEMANTIC_CATEGORIES('SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL');
```

Results:

EXTRACT_SEMANTIC_CATEGORIES('SNOWPIPE.DATA_CLASS.SAMPLE_DATA_TBL')
{ "AGE": { "alternates": [], "recommendation": { "confidence": "HIGH", "coverage": 1, "details": [], "privacy_category": "QUASI_IDENTIFIER", "semantic_category": "Age" } }, "CC": { "alternates": [], "recommendation": { "confidence": "HIGH", "coverage": 1, "details": [], "privacy_category": "CreditCard", "semantic_category": "CreditCard" } }, "ID": { "alternates": [], "recommendation": { "confidence": "HIGH", "coverage": 1, "details": [], "privacy_category": "QuasiIdentifier", "semantic_category": "QuasiIdentifier" } }, "SSN": { "alternates": [], "recommendation": { "confidence": "HIGH", "coverage": 1, "details": [], "privacy_category": "SocialSecurityNumber", "semantic_category": "SocialSecurityNumber" } }}

INNOVATION SOFTWARE LEARN. EMPOWER. INNOVATE.

# DATA GOVERNANCE - MASKING

Snowflake supports masking policies as a schema-level object to protect sensitive data from unauthorized access while allowing authorized users to access sensitive data at query runtime. This means that sensitive data in Snowflake is not modified in an existing table (i.e. no static masking). Rather, when users execute a query in which a masking policy applies, the masking policy conditions determine whether unauthorized users see masked, partially masked, obfuscated, or tokenized data.

Authorized role (i.e. SUPPORT)

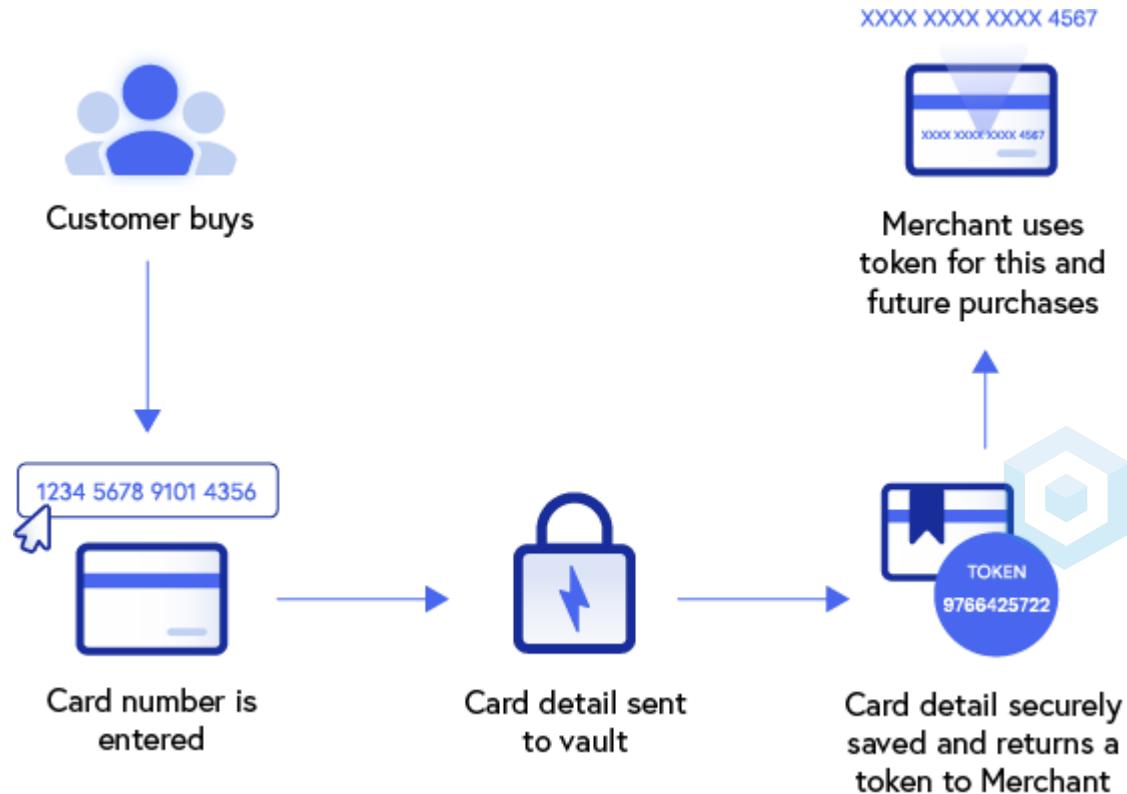
ID	Phone	SSN
101	408-123-5534	387-78-3456
102	510-334-3564	226-44-8908
103	214-553-9787	359-9987-0098

Unauthorized role (i.e. ANALYST)

ID	Phone	SSN
101	***-**-5534	*****
102	***-**-3564	*****
103	***-**-9787	*****

# DATA GOVERNANCE - TOKENIZATION

External Tokenization enables accounts to tokenize data before loading it into Snowflake and detokenize the data at query runtime. Tokenization is the process of removing sensitive data by replacing it with an undecipherable token. External Tokenization makes use of masking policies with [external functions](#).



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# ENCRYPTION VS TOKENIZATION

Encryption essentially means scrambling sensitive data that must then be decrypted with a unique key in order to be read. If the key and algorithm is identified, the decrypted data is accessible.

Tokenization involves swapping sensitive data for a token that must then be presented in order to retrieve the data. If the token is identified, the source data is not accessible



# MASKING/TOKENIZATION

## Step 1: Grant Masking Policy Privileges to Custom Role

A security or privacy officer should serve as the masking policy administrator (i.e. custom role: MASKING\_ADMIN) and have the privileges to define, manage, and apply masking policies to columns.

The following example creates the MASKING\_ADMIN role and grants masking policy privileges to that role.

Create a masking policy administrator custom role:

```
use role useradmin;
CREATE ROLE masking_admin;
```

Grant privileges to masking\_admin role:

```
use role securityadmin;
GRANT CREATE MASKING POLICY on SCHEMA <db_name.schema_name> to ROLE masking_admin;
GRANT APPLY MASKING POLICY on ACCOUNT to ROLE masking_admin;
```

Allow table\_owner role to set or unset the ssn\_mask masking policy (optional):

```
GRANT APPLY ON MASKING POLICY ssn_mask to ROLE table_owner;
```

Where:

- <db\_name.schema\_name>

Specifies the identifier for the schema for which the privilege should be granted.



# MASKING/TOKENIZATION

## Step 2: Grant the Custom Role to a User

Grant the `MASKING_ADMIN` custom role to a user serving as the security or privacy officer.

```
use role useradmin;
grant role masking_admin to user jsmith;
```



# MASKING/TOKENIZATION

## Step 3: Create a Masking Policy

In this representative example, users with the ANALYST custom role see the detokenized email values. Users without the ANALYST custom role see the tokenized values.

The external function to detokenize email values is de\_email().

```
-- create masking policy

create or replace masking policy email_de_token as (val string) returns string ->
  case
    when current_role() in ('ANALYST') then de_email(val)
    else val
  end;
```



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# MASKING/TOKENIZATION

## Step 4: Apply the Masking Policy to a Table or View Column

These examples assume that a masking policy is not applied to the table column when the table is created and the view column when the view is created. You can optionally apply a masking policy to a table column when you create the table with a [CREATE TABLE](#) statement or a view column with a [CREATE VIEW](#) statement.

Execute the following statements to apply the policy to a table column or a view column.

```
-- apply masking policy to a table column  
  
alter table if exists user_info modify column email set masking policy email_de_token;  
  
-- apply the masking policy to a view column  
  
alter view user_info_v modify column email set masking policy email_de_token;
```

# MASKING/TOKENIZATION

## Step 6: Query Data in Snowflake

Execute two different queries in Snowflake, one query with the ANALYST role and another query with a different role, to verify that users without the ANALYST role see a full mask.

```
-- using the ANALYST role
```

```
USE ROLE analyst;  
SELECT email FROM user_info; -- should see plain text value
```

```
-- using the PUBLIC role
```

```
USE ROLE PUBLIC;  
SELECT email FROM user_info; -- should see full data mask
```



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# MASKING/TOKENIZATION – BEST PRACTICES

Synchronizing systems. On AWS, it is helpful to synchronize users and roles in your organization's identity provider (IdP) with Snowflake and Protegrity. If users and roles are not synchronized, there can be unexpected behaviors, error messages, and complex troubleshooting regarding external functions, API integrations, masking policies, and tokenization policies. One option is to use SCIM to keep users and roles synchronized with your IdP and Snowflake.

Root cause for error(s). Since External Tokenization requires coordinating multiple systems (e.g. IdP, Snowflake, Protegrity, AWS, Azure, GCP), always verify the privileges, current limitations, external functions, API integration, masking policies, and the columns that have masking policies for External Tokenization in Snowflake.

# DATA GOVERNANCE - ROW ACCESS POLICIES

Snowflake supports row-level security through the use of row access policies to determine which rows to return in the query result. The row access policy can be relatively simple to allow one particular role to view rows, or be more complex to include a mapping table in the policy definition to determine access to rows in the query result.



# DATA GOVERNANCE - ROW ACCESS POLICIES

We first want to create sample roles and users in Snowflake

Note – You don't need to create multiple users, but you can assign multiple roles to your user

```
SNOWTEST.PUBLIC ▾      Settings ▾  
1 USE ROLE ACCOUNTADMIN;  
2  
3 CREATE OR REPLACE DATABASE SNOWTEST;  
4 CREATE OR REPLACE SCHEMA SNOWTEST.PUBLIC;  
5  
6 CREATE OR REPLACE USER user1 PASSWORD = 'user1' MUST_CHANGE_PASSWORD = FALSE;  
7 CREATE OR REPLACE USER user2 PASSWORD = 'user2' MUST_CHANGE_PASSWORD = FALSE;  
8 CREATE OR REPLACE USER superadmin PASSWORD = 'superadmin' MUST_CHANGE_PASSWORD = FALSE;  
9 CREATE OR REPLACE ROLE ROLE1;  
10 CREATE OR REPLACE ROLE ROLE2;  
11 CREATE OR REPLACE ROLE ROLE3;  
12 CREATE OR REPLACE ROLE SUPERADMIN;  
13  
14 GRANT ROLE SUPERADMIN TO USER KEVERETTEXIT;  
15 GRANT ROLE ROLE1 TO USER KEVERETTEXIT;  
16 GRANT ROLE ROLE2 TO USER KEVERETTEXIT;  
17 GRANT ROLE ROLE3 TO USER KEVERETTEXIT;
```



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# DATA GOVERNANCE - ROW ACCESS POLICIES

Next, recreate the sample data set

For this example, we will have a table with state, SSN, AGE and CC information.

We also need to create a mapping table for each role and value we want to map on (in this case, state)

```
CREATE OR REPLACE TABLE SNOWTEST.PUBLIC.SAMPLE_DATA_TBL(
    STATE VARCHAR(2)
    , SSN VARCHAR(11)
    , AGE NUMERIC
    , CC VARCHAR(19)
);

INSERT INTO SNOWTEST.PUBLIC.SAMPLE_DATA_TBL VALUES ('KS','234-45-6477',27,'4053 0495 0394 0494'), ('TX','234-85-6477',67,'4653 0495 0394 0494'),
('TX','235-45-6477',44,'4053 0755 0394 0494'), ('MD','234-85-6477',81,'4873 0495 0394 4094'), ('CA','234-85-0877',18,'4653 0495 0084 0494');

CREATE OR REPLACE TABLE SNOWTEST.PUBLIC.MAPPING (
    ROLE_ENTITLED varchar, STATE varchar
);

INSERT INTO SNOWTEST.PUBLIC.MAPPING VALUES ('ROLE1','TX'),('ROLE1','KS'),('ROLE1','MD'),('ROLE1','CA'),('ROLE1','TX'),('ROLE1','NA'),('ROLE3','TX');
```



# DATA GOVERNANCE - ROW ACCESS POLICIES

Next, create Row Access Policy between roles and filter mapping

Grant the roles created to the table for access

```
CREATE ROW ACCESS POLICY SNOWTEST.PUBLIC.TEST_POLICY AS
(state_filter varchar) RETURNS BOOLEAN ->
CURRENT_ROLE() = 'SUPERADMIN'
OR EXISTS (
    SELECT 1 FROM SNOWTEST.PUBLIC.MAPPING
    WHERE STATE = state_filter
    AND ROLE_ENTITLED = CURRENT_ROLE());
ALTER TABLE SNOWTEST.PUBLIC.SAMPLE_DATA_TBL ADD ROW ACCESS POLICY SNOWTEST.PUBLIC.TEST_POLICY ON (STATE);
USE ROLE ACCOUNTADMIN;
GRANT SELECT ON SNOWTEST.PUBLIC.SAMPLE_DATA_TBL TO ROLE ROLE1;
GRANT SELECT ON SNOWTEST.PUBLIC.SAMPLE_DATA_TBL TO ROLE ROLE2;
GRANT SELECT ON SNOWTEST.PUBLIC.SAMPLE_DATA_TBL TO ROLE ROLE3;
GRANT SELECT ON SNOWTEST.PUBLIC.SAMPLE_DATA_TBL TO ROLE SUPERADMIN;
```



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# DATA GOVERNANCE - ROW ACCESS POLICIES

Test by using each role created

You will only see data where the role has access

```
//  
78  USE ROLE ROLE2;  
79  SELECT CURRENT_ROLE();  
80  SELECT * FROM SNOWTEST.PUBLIC.SAMPLE_DATA_TBL;  
81  
82  USE ROLE ROLE3;  
83  SELECT CURRENT_ROLE();  
84  SELECT * FROM SNOWTEST.PUBLIC.SAMPLE_DATA_TBL;  
85  
86  USE ROLE ACCOUNTADMIN;
```

↳ Results    ↙ Chart

	STATE	SSN	AGE	CC
1	TX	234-85-6477	67	4653 0495 0394 0494
2	TX	235-45-6477	44	4053 0755 0394 0494



INNOVATION  
SOFTWARE

LEARN. EMPOWER. INNOVATE.

# DATA GOVERNANCE - ACCESS HISTORY

Access History in Snowflake refers to when the user query reads data and when the SQL statement performs a data write operation, such as INSERT, UPDATE, and DELETE along with variations of the COPY command, from the source data object to the target data object. The user access history can be found by querying the Account Usage [ACCESS HISTORY](#) view.

# DATA GOVERNANCE - DEPENDENCIES

An object dependency means that in order to operate on an object, the object that is being operated on must reference metadata for itself or reference metadata for at least one other object. Snowflake tracks object dependencies in the Account Usage view OBJECT\_DEPENDENCIES..

# POP QUIZ:

## Snowflake Architecture



Referential integrity constraints are enforced?

- A: True
- B: False

# POP QUIZ:

## Snowflake Architecture



Referential integrity constraints are enforced?

A: True

B: False

Referential integrity constraints are informational and not enforced. The only constraint that is enforced is “NOT NULL”



# POP QUIZ:

## Snowflake Architecture



What are two examples of Snowflake Continuous Data Protection:

- A: Database backups
- B: Time travel
- C: RBAC
- D: Fail-safe
- E: Cloning

# POP QUIZ:

## Snowflake Architecture



What are two examples of Snowflake Continuous Data Protection:

- A: Database backups
- B: Time travel
- C: RBAC
- D: Fail-safe
- E: Cloning

# POP QUIZ:

## Snowflake Architecture



What are the types of views that Snowflake supports:

- A: Materialized
- B: Un-Materialized
- C: Re-materialized
- D: Non-materialized

# POP QUIZ:

## Snowflake Architecture



What are the types of views that Snowflake supports:

- A: Materialized
- B: Un-Materialized
- C: Re-materialized
- D: Non-materialized

# POP QUIZ:

## Snowflake Architecture



What table types bypass CDP:

- A: Transient
- B: Mobile
- C: Materialized
- D: Temporary

# POP QUIZ:

## Snowflake Architecture



What table types bypass CDP:

- A: Transient
- B: Mobile
- C: Materialized
- D: Temporary

# POP QUIZ:

## Snowflake Architecture



Secure views do not preserve the view optimizations that Snowflake has on normal views :

- A: True
- B: False

# POP QUIZ:

## Snowflake Architecture



Secure views do not preserve the view optimizations that Snowflake has on normal views :

A: True

B: False

Snowflake optimizations for views require access to the underlying data in the base tables for the view.

Secure views do not utilize these optimizations, ensuring that users have no access to the underlying data.

# POP QUIZ:

## Snowflake Architecture



Clones have the same privileges as the original object?

- A: True
- B: False

# POP QUIZ:

## Snowflake Architecture



Clones have the same privileges as the original object?

A: True

B: False

A cloned object does not retain any granted privileges on the source object itself (i.e. clones do not automatically have the same privileges as their sources). A system administrator or the owner of the cloned object must explicitly grant any required privileges to the newly-created clone.

# POP QUIZ:

## Snowflake Architecture



You can get information about the type of columns in a table with the Information Schema?

- A: True
- B: False

# POP QUIZ:

## Snowflake Architecture



You can get information about the type of columns in a table with the Information Schema?

A: True

B: False

Information Schema enables users the ability to acquire metadata about their objects.



# POP QUIZ:

## Snowflake Architecture



Which one of these Data Governance roles is accountable for building the data warehouse?

- A: Data governance council
- B: Data managers
- C: Data owners
- D: Data stewards

# POP QUIZ:

## Snowflake Architecture



Which one of these Data Governance roles is accountable for building the data warehouse?

- A: Data governance council
- B: Data managers
- C: Data owners
- D: Data stewards

Creates database systems that meet an organization's needs



# POP QUIZ:

## Snowflake Architecture



Tokenization uses a key and algorithm to scramble and encrypt data, send the key and encrypted data to another system to be deciphered.

- A: TRUE
- B: FALSE

# POP QUIZ:

## Snowflake Architecture



Tokenization uses a key and algorithm to scramble and encrypt data, send the key and encrypted data to another system to be deciphered.

A: TRUE

B: FALSE

Tokenization is the process of removing sensitive data by replacing it with an undecipherable token



# POP QUIZ:

## Snowflake Architecture



What function enables Snowflake to access the schema of a table and determine sensitive data:

- A: ACCOUNT\_USAGE
- B: EXTRACT\_SCHEMA\_CATEGORIES
- C: EXTRACT\_SEMANTIC\_CATEGORIES
- D: DOES NOT EXIST

# POP QUIZ:

## Snowflake Architecture



What function enables Snowflake to access the schema of a table and determine sensitive data columns:

- A: ACCOUNT\_USAGE
- B: EXTRACT\_SCHEMA\_CATEGORIES
- C: EXTRACT\_SEMANTIC\_CATEGORIES
- D: DOES NOT EXIST

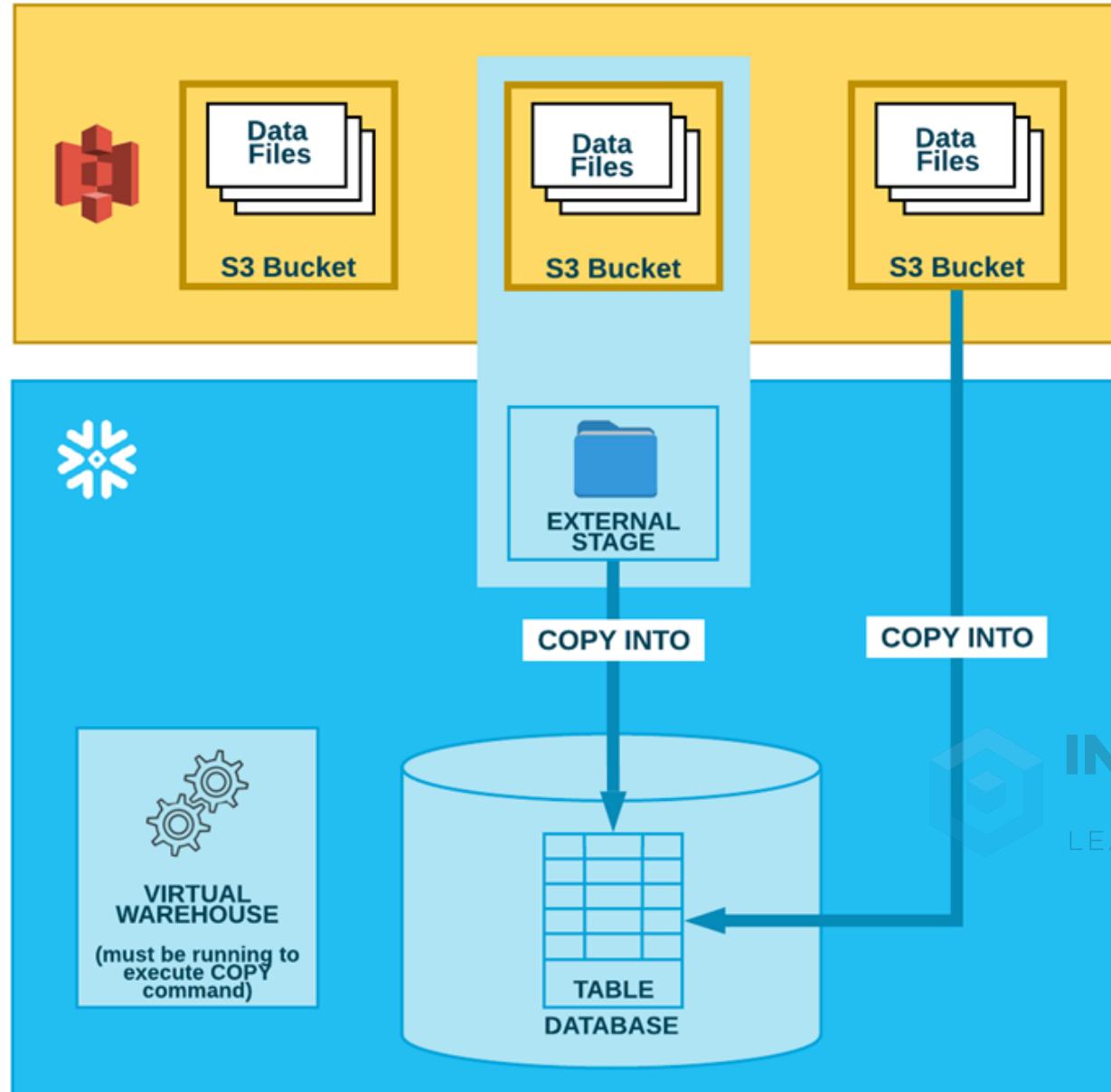
EXTRACT\_SEMANTIC\_CATEGORIES will scan the table and attempt to classify each column according



# Data Movement



# DATA LOADING



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# DATA LOADING TOPICS

- Supported File Locations
  - External Stages
  - Internal Stages
- Bulk vs Continuous Loading
  - Bulk Loading Using the COPY Command
  - Continuous Loading Using Snowpipe
- Loading Data from Apache Kafka Topics
- Alternatives to Loading Data
  - External Tables (Data Lake)



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# DATA LOADING STAGES

Snowflake refers to the location of data files in cloud storage as a **stage**.

The **COPY INTO** commands are used for both bulk and continuous data loads (i.e. Snowpipe) into stages

- **External Stage** - cloud storage accounts managed by your business entity – Azure/GCP/AWS
- **Internal Stage** - cloud storage contained in your Snowflake account



# DATA LOADING: COPY

The **COPY INTO <table>** loads data from staged files to an existing table. The files must already be staged in one of the following locations:

- Named internal stage (or table/user stage). Files can be staged using the PUT command.
- Named external stage that references an external location (Amazon S3, Google Cloud Storage, or Microsoft Azure).
- External location (Amazon S3, Google Cloud Storage, or Microsoft Azure).

# DATA UNLOADING: COPY

The **COPY INTO <location>** unloads data from a table (or query) into one or more files in one of the following locations:

- Named internal stage (or table/user stage). The files can then be downloaded from the stage/location using the GET command.
- Named external stage that references an external location (Amazon S3, Google Cloud Storage, or Microsoft Azure).
- External location (Amazon S3, Google Cloud Storage, or Microsoft Azure).

# DATA LOADING: INSERT

Using a single `INSERT` command, you can insert multiple rows into a table by specifying additional sets of values separated by commas in the `VALUES` clause.

For example, the following clause would insert 3 rows in a 3-column table, with values 1, 2, and 3 in the first two rows and values 2, 3, and 4 in the third row:

```
values ( 1, 2, 3 ) , ( 1, 2, 3 ) , ( 2, 3, 4 )
```

To use the `OVERWRITE` option on `INSERT`, you must use a role that has `DELETE` privilege on the table because `OVERWRITE` will delete the existing records in the table.

Some expressions cannot be specified in the `VALUES` clause.

# DATA LOADING: GET

Downloads data files from one of the following Snowflake stages to a local directory/folder on a client machine:

- Named internal stage.
- Internal stage for a specified table.
- Internal stage for the current user.

Typically, this command is executed after using the `COPY INTO <location>` command to unload data from a table into a Snowflake stage.

# DATA LOADING: PUT

Uploads (i.e. stages) data files from a local directory/folder on a client machine to one of the following Snowflake stages:

- Named internal stage
  - Internal stage for a specified table
  - Internal stage for the current user
- 
- Once files are staged, the data in the files can be loaded into a table using the COPY INTO <table> command.

# DATA LOADING: VALIDATE

Validates the files loaded in a past execution of the COPY INTO <table> command and returns all the errors encountered during the load, rather than just the first error.

The validation returns no results for COPY statements that specify ON\_ERROR = ABORT\_STATEMENT (default value).

Validation fails if:

- The current user does not have access to table\_name.
- The current user is not the user who executed query\_id and does not have access control privileges on this user.
- If new files have been added to the stage used by query\_id since the load was executed, the new files added are ignored during the validation.

# FILE FORMATS

A named file format object provides a convenient means to store all of the format information required for loading data from files into tables.

Execute CREATE FILE FORMAT to create a file format that you can reference throughout your project or data warehouse usage.

This step is optional, but is recommended when you plan to load large numbers of files of a specific format.



# CSV FILE FORMATS

To create a CSV file format:

TYPE - defaults to CSV

FIELD\_DELIMITER – defaults to a comma

SKIP\_HEADER an integer value defining the number of lines in the header. The COPY command skips these lines when loading data. The default value is 0.

```
create or replace file format mycsvformat
  type = 'CSV'
  field_delimiter = '|'
  skip_header = 1;
```



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# JSON FILE FORMATS

To create a JSON file format:

TYPE - defaults to JSON

STRIP\_OUTER\_ARRAY = TRUE. Instructs the JSON parser to remove the root brackets [ ].

```
create or replace file format myjsonformat
  type = 'JSON'
  strip_outer_array = true;
```



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# CREATE STAGES

Create stages using the **CREATE STAGE** command.

A named external stage is a database object created in a schema.

This object stores the **URL** to files in cloud storage, the **settings** used to access the cloud storage account, and convenience settings such as the **options** that describe the format of staged files.

# CREATE STAGES

Execute CREATE STAGE to create a named internal stage. This step is recommended when you plan to load data files regularly from the same source.

Our example CREATE STAGE commands create internal stages that specify the file formats that we showed for CSV and JSON formats.

COPY commands default to the file formats that are specified when we use CREATE STAGE and do not need to be specified on the COPY command.

# CREATE CSV-FORMATTED STAGE

The following example creates an internal stage named my\_csv\_stage.

Parameter values that aren't specified use the default values (DATE\_FORMAT = AUTO, COMPRESSION = AUTO, etc.).

```
create or replace stage my_csv_stage  
    file_format = mycsvformat;
```



# CREATE JSON-FORMATTED STAGE

The following example creates an internal stage named my\_json\_stage.

Parameter values that aren't specified use the default values (DATE\_FORMAT = AUTO, COMPRESSION = AUTO, etc.).

```
create or replace stage my_json_stage  
file_format = myjsonformat;
```



# STAGE FOLDER STANDARDIZATION

Normally a load will include a set of data files. Those files may be packaged in zip or gzip archives. Normally we'd unpack archives to our load file system, optimally setting a standard. That type of folder standardization allows teams to understand data project handling by convention.

We could unpack data file archives to any location. To follow our recommended best practice, we'll set standard directories to be referenced. This requires specifics based on platform, for the most commonly used client/server platforms that would equate to:

**Linux or macOS:** /tmp/load.

**Windows:** C:\temp\load.

# EXTERNAL STAGES

Loading data from any of the following cloud storage services is supported regardless of the cloud platform that hosts your Snowflake account:

- Amazon S3
- Google Cloud Storage
- Microsoft Azure

Upload (i.e. stage) files to your cloud storage account using the tools provided by the cloud storage service.



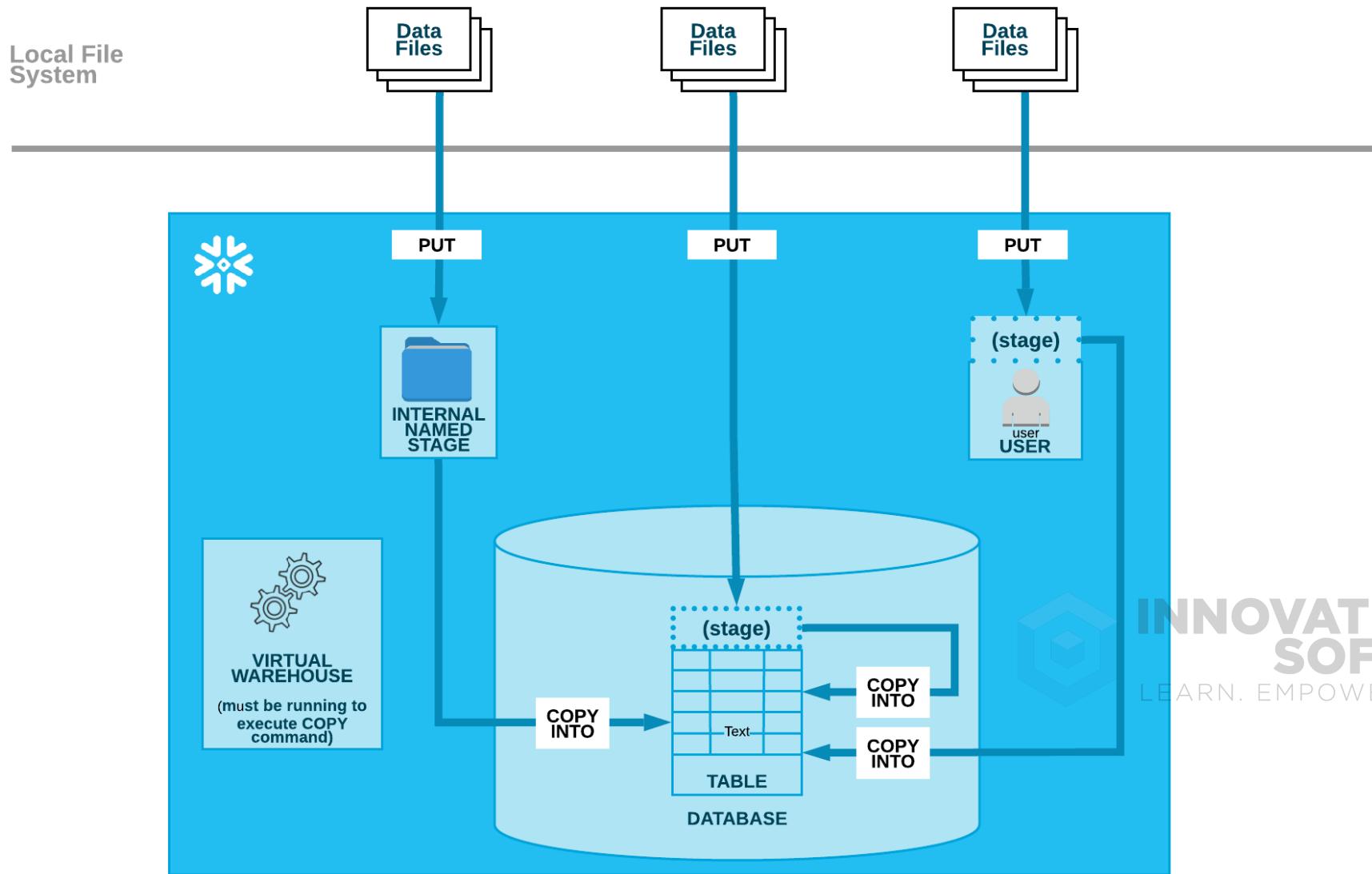
# BULK LOADING

Bulk Loading enables loading batches of data from files already available in cloud storage.

Alternatively we can copy (i.e. staging) data files from a local machine to an internal (i.e. Snowflake) cloud storage location before loading the data into tables using the COPY command.

Bulk loading relies on user-provided virtual warehouses, which are specified in the COPY statement. Users are required to size the warehouse appropriately to accommodate expected loads.

# BULK LOADING FROM LOCAL FILE



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# LOAD TRANSFORMATIONS

Snowflake supports transforming data while loading it into a table using the COPY command.

Options include:

- Column reordering
- Column omission
- Casts
- Truncating text strings that exceed the target column length

There is no requirement for your data files to have the same number and ordering of columns as your target table.



# DATA WRANGLING

Data **wrangling** —also called data **cleansing**, data **transformation**, data **remediation**, or data **munging** —refers to a variety of processes designed to transform raw data into more readily used formats.

The exact methods differ from project to project depending on the data you're leveraging and the goal you're trying to achieve.

There are many tools that assist in the wrangling journey including ad-hoc query tools like Facebook Presto and AWS Athena.

# DATA LOAD PROCESS

- Step 1. Create File Format Objects
- Step 2. Create Stage Objects
- Step 3. Stage the Data Files
- Step 4. Verify the Staged Files
- Step 5. Copy Data into the Target Tables
- Step 6. Resolve Data Load Errors Related to Data Issues
- Step 7. Validate the Loaded Data
- Step 8. Remove the Successfully Loaded Data Files

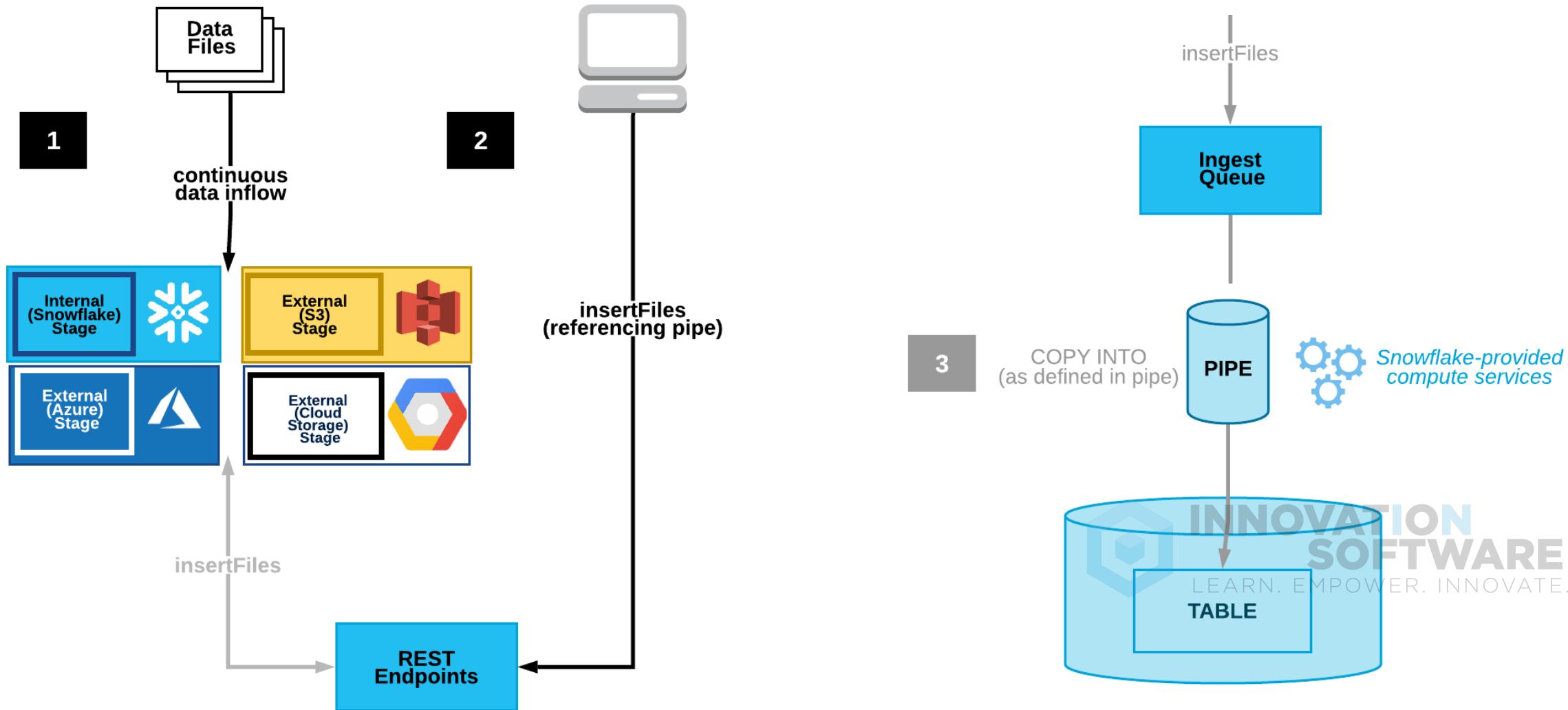


# CONTINUOUS LOADING

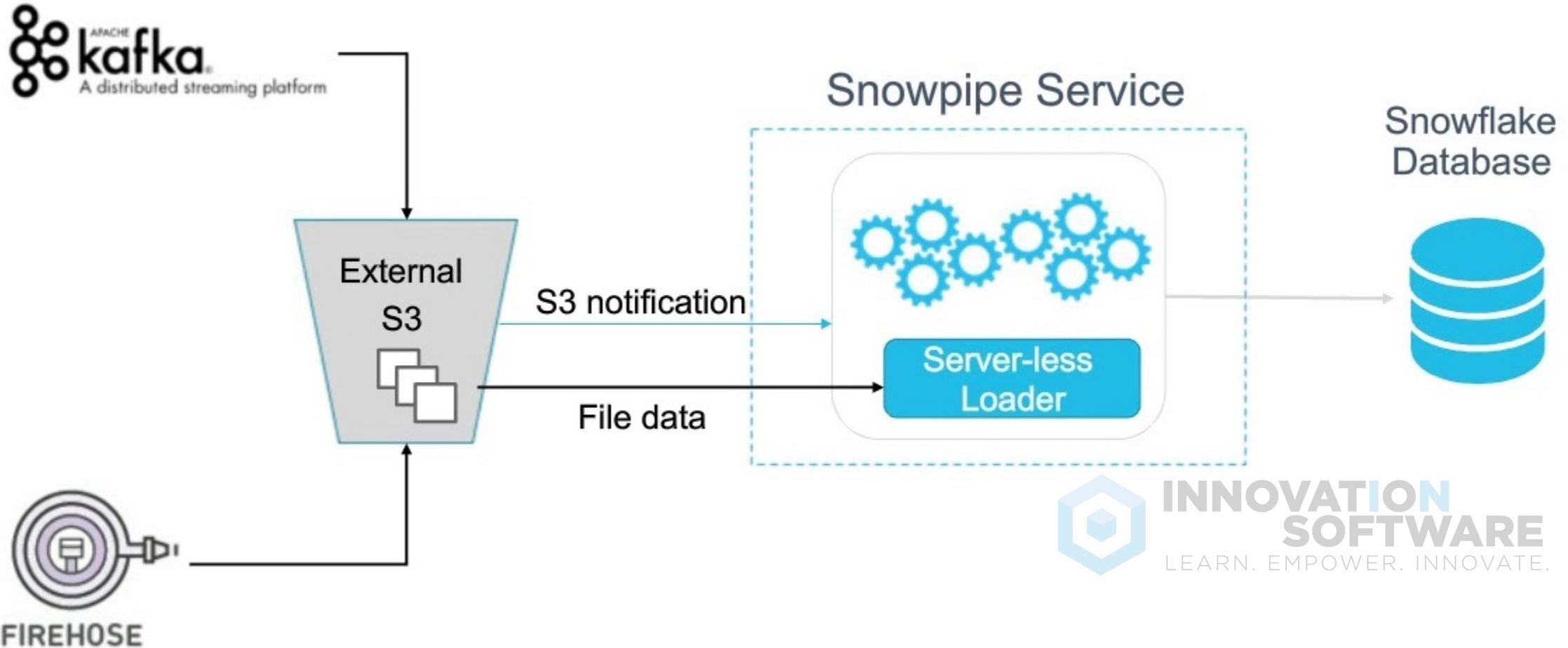
**Snowpipe** is Snowflake's **continuous data ingestion** service. Snowpipe loads data within minutes after files are added to a stage and submitted for ingestion.

With Snowpipe's **serverless** compute model, Snowflake manages load capacity, ensuring optimal compute resources to meet demand. In short, Snowpipe provides a "pipeline" for **loading data in micro-batches** as soon as it's available.

# CONTINUOUS LOADING REST



# CONTINUOUS LOADING S3



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# CONFIGURING SNOWPIPE 4 REST

- Create a named stage object where your data files will be staged. Snowpipe supports both internal (Snowflake) stages and external stages, i.e. S3 buckets.
- Create a pipe object using **CREATE PIPE**.
- Configure security for the user who will execute the continuous data load. If you plan to restrict Snowpipe data loads to a single user, you only need to configure key pair authentication for the user once. After that, you only need to grant access control privileges on the database objects used for each data load.
- Install a client SDK (Java or Python) for calling the Snowpipe public REST endpoints.

# DATA UNLOADING

Similar to data loading, Snowflake supports bulk export (i.e. unload) of data from a database table into flat, delimited text files. The unloading process involves the following topics:

Bulk Unloading Process

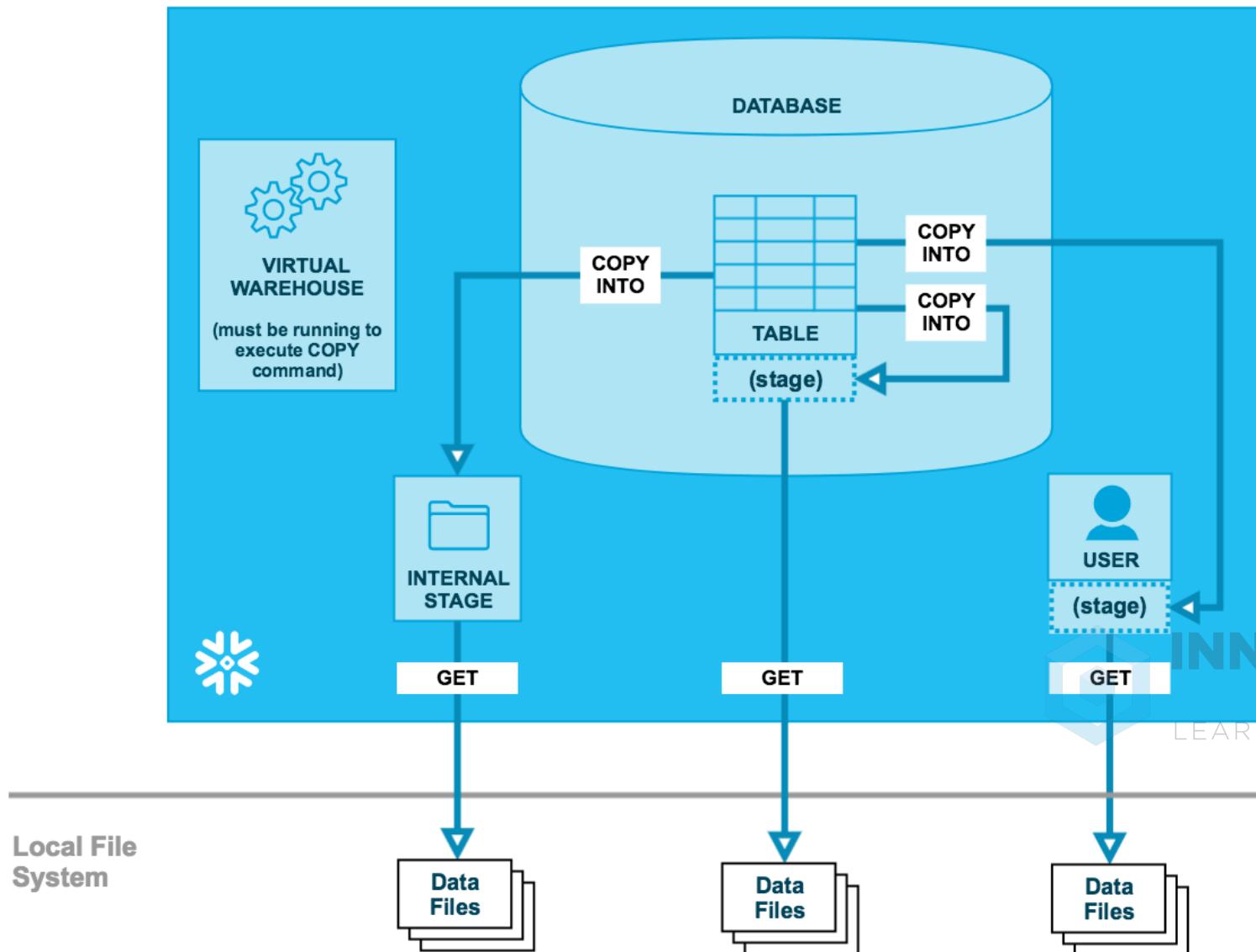
Bulk Unloading Using Queries

Bulk Unloading into Single or Multiple Files

Partitioned Data Unloading

Tasks for Unloading Data Using the COPY Command

# DATA UNLOADING INTO STAGE



INNOVATION  
SOFTWARE

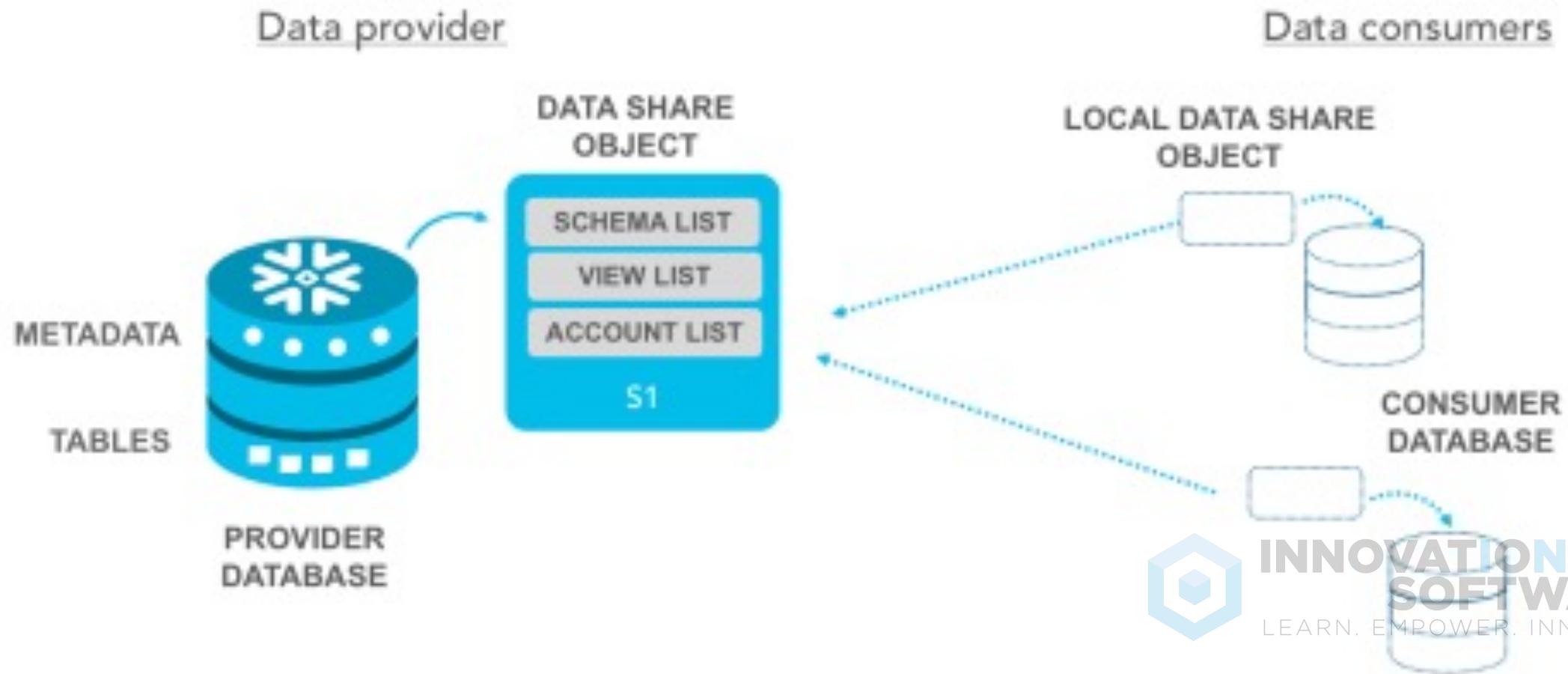
LEARN. EMPOWER. INNOVATE.

# DATA UNLOADING PROCESS

The process for unloading data into files is the same as the loading process, except in reverse:

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

# DATA SHARING



# DATA SHARING

Data sharing is the act of providing access to data — both within an enterprise and between enterprises that have determined they have valuable assets to share.

The organization that makes its data available, or shares its data, is a ***data provider***. The organization that wants to use the shared data is a ***data consumer***. Any organization can be a data provider, a data consumer, or both.



# DATA SHARING IN SNOWFLAKE

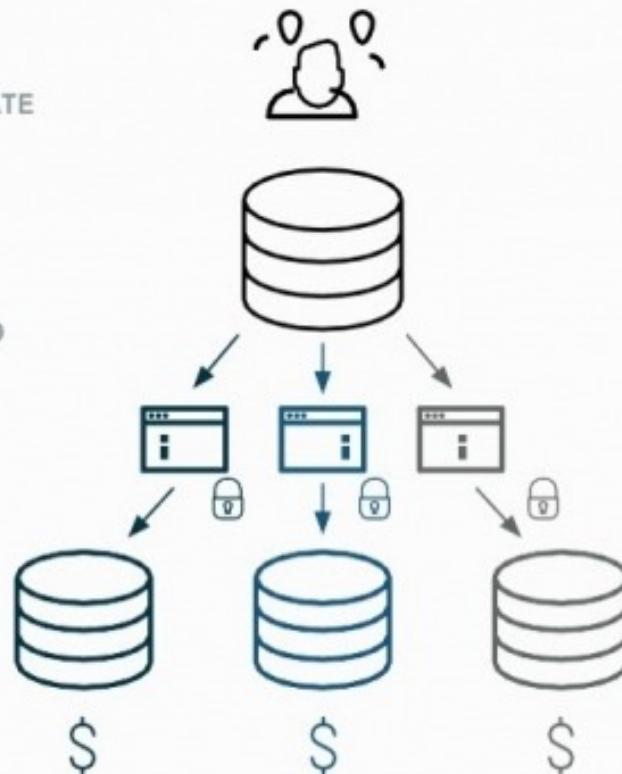
## Distributing Data with Traditional

SIGNIFICANT CORPORATE

MANUAL UPDATES AND

REPETITIVE  
OVERHEAD FOR

DUPLICATIVE

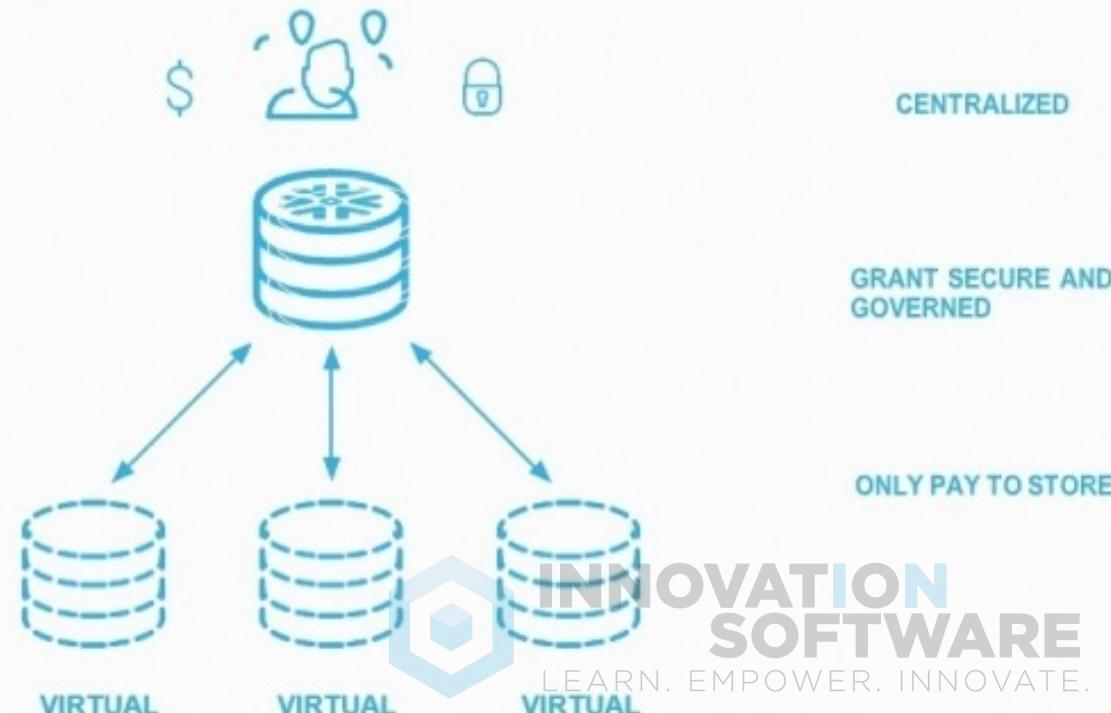


## Sharing Data Across Business Units with

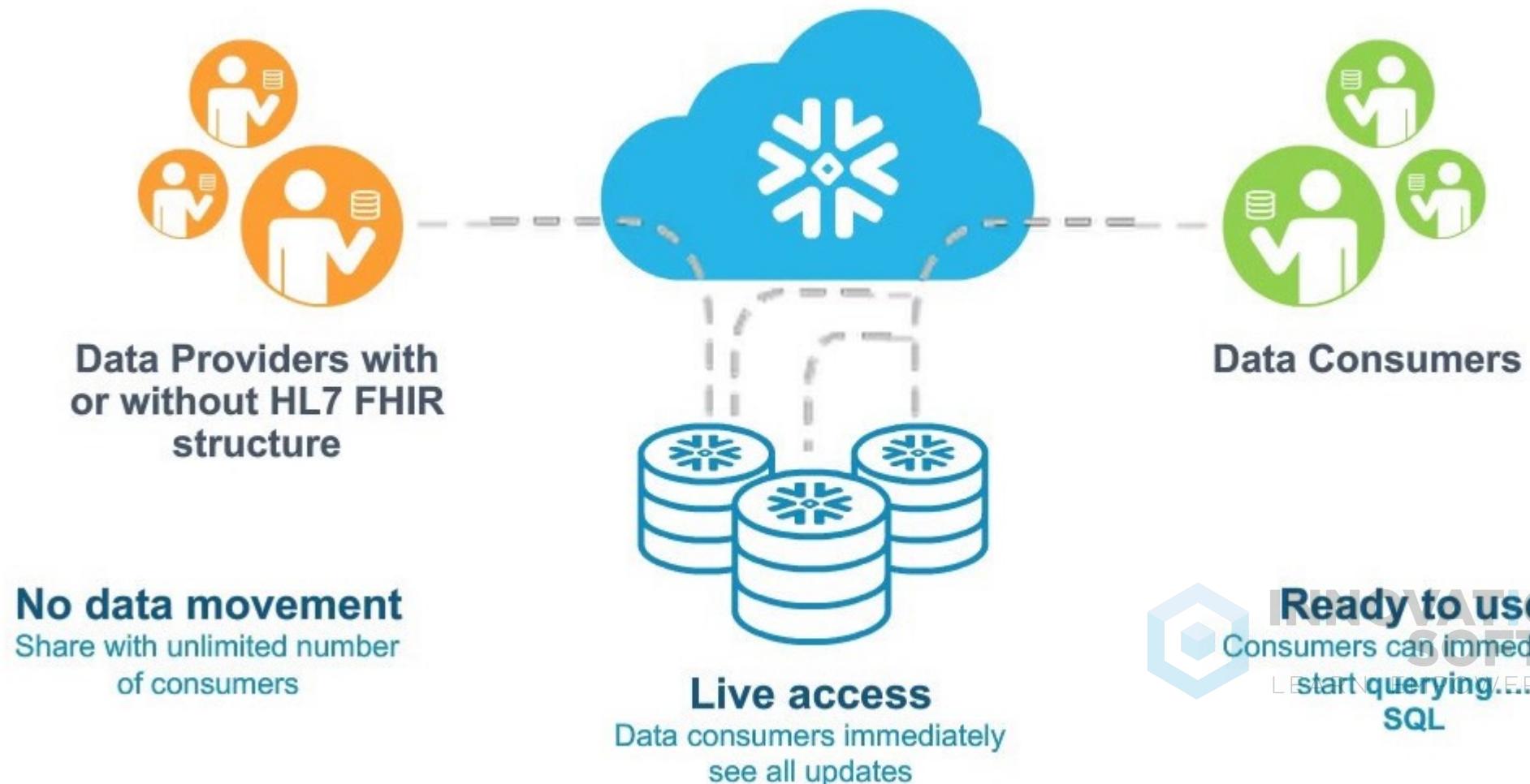
CENTRALIZED

GRANT SECURE AND  
GOVERNED

ONLY PAY TO STORE



# DATA SHARING VS MOVING



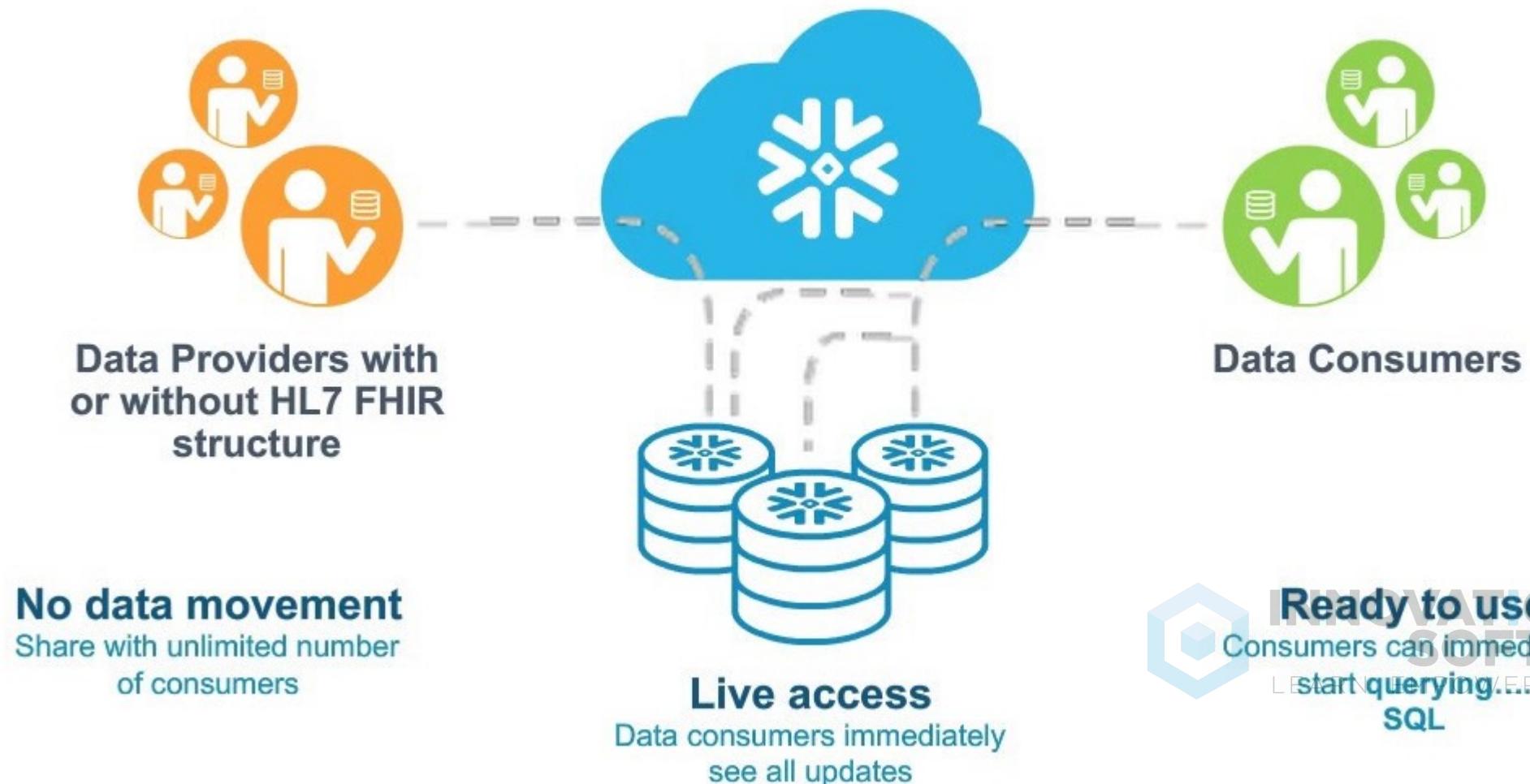
**Ready to use**  
ON SOFTWARE  
Consumers can immediately start querying...  
LEARN MORE. INNOVATE.  
SQL

# DATA SHARING EVOLUTION

Most organizations that embark on a data sharing journey follow a familiar progression:

1. Internal collaboration
2. Business insights
3. Customer analytics
4. Advanced analytics
5. Data services
6. Data exchange

# DATA SHARING VS MOVING



**Ready to use**  
ON SOFTWARE  
Consumers can immediately start querying...  
LEARN MORE. INNOVATE.  
SQL

# DATA LINAGE

Data lineage refers to the ability to trace the origin, movement, and transformation of data throughout its lifecycle. It involves understanding the path of data from its source systems, through various processes and transformations, to its final destination.

Snowflake automatically captures and maintains data lineage information without requiring manual intervention. This data is available to Snowflake users through a special view called ACCESS\_HISTORY, which holds the history of tables, views, and columns for up to one year.

# DATA LINAGE

1. Create test table
2. Query and Add Data

```
CREATE TABLE DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST
(
    id int,
    message text
);
```

```
select * from DATA_LINEAGE_TEST;
insert into DATA_LINEAGE_TEST VALUES(1, 'HELLO'), (2, 'GOODBYE');
select * from DATA_LINEAGE_TEST;
update DATA_LINEAGE_TEST set MESSAGE = 'BONJOUR' where ID = 1;
select * from DATA_LINEAGE_TEST;
```



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# DATA LINAGE

1. Check the Linage in the Access\_History

```
select * from SNOWFLAKE.ACCOUNT_USAGE.ACCESS_HISTORY
where base_objects_accessed[0].objectName = 'DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST'
    or objects_modified[0].objectName = 'DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST'
order by query_start_time;
```

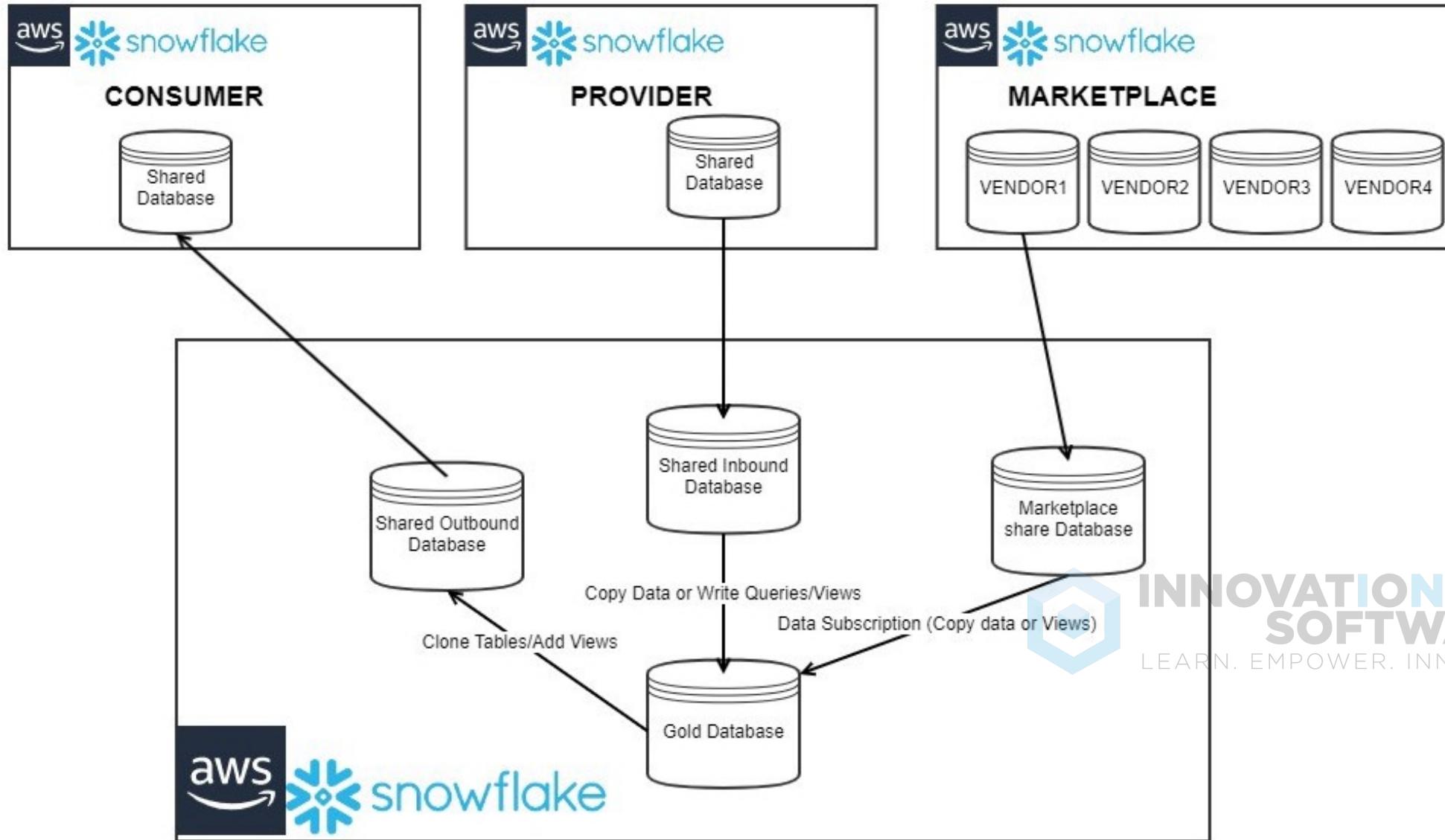
Results:

QUERY_START_TIME	USER_NAME	DIRECT_OBJECTS_ACCESSED
2023-06-19 09:51:56.124 -0700	JDELISSI	[ { "columns": [ { "columnId": 1, "columnName": "ID" } ], "objectName": "DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST" } ]
2023-06-19 09:53:06.959 -0700	JDELISSI	[]
2023-06-19 09:53:25.318 -0700	JDELISSI	[ { "columns": [ { "columnId": 1, "columnName": "ID" } ], "objectName": "DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST" } ]
2023-06-19 09:54:42.145 -0700	JDELISSI	[ { "columns": [ { "columnId": 1, "columnName": "ID" } ], "objectName": "DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST" } ]
2023-06-19 09:54:46.726 -0700	JDELISSI	[ { "columns": [ { "columnId": 1, "columnName": "ID" } ], "objectName": "DATA_LINEAGE.PHDATA.DATA_LINEAGE_TEST" } ]



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# BEST PRACTICES



# POP QUIZ:

## Data Movement



Which of the following are data loading SQL commands?

- A: VALIDATE
- B: PUT
- C: GET
- D: INSERT
- E: COPY

# POP QUIZ:

## Data Movement



Which of the following are data loading SQL commands?

- A: VALIDATE
- B: PUT
- C: GET
- D: INSERT
- E: COPY

# POP QUIZ:

## Data Movement



Worksheet context has defaults available for which options?

- A: Database
- B: Schema
- C: Warehouse
- D: Role
- E: All of the above



# POP QUIZ:

## Data Movement



Worksheet context has defaults available for which options?

- A: Database
- B: Schema
- C: Warehouse
- D: Role
- E: All of the above



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# Objects and Commands



# QUERY CONSTRUCTS

Snowflake supports standard SQL, including a subset of ANSI SQL:1999 and the SQL:2003 analytic extensions.

Snowflake also supports common variations for a number of commands where those variations do not conflict with each other.



# QUERY EDITORS

Following are the list of commonly used Snowflake SQL editor tools.

- Snowflake SQL (SnowSQL)
- Snowflake Web UI
- SQuirrel SQL Client
- SQL Workbench
- DBeaver

# SNOWFLAKE QUERY EXECUTION

There are two interactive Snowflake tools that can be used to connect to a Snowflake instance:

SnowSQL command-line utility (CLI)  
Snowflake WebUI.

The Snowflake WebUI is the most common way to connect to a Snowflake instance, execute queries, and perform administrative functions such as creating new database objects, managing virtual warehouses, managing security, and reviewing the costs associated with your instance.

# SNOWFLAKE WEB UI

The screenshot shows the Snowflake Web UI interface. At the top, there's a header bar with a back arrow, forward arrow, refresh icon, and a URL bar displaying <https://lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet>. Below the header is a banner with the text "Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact support". The main navigation bar includes icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (which is selected and highlighted in grey), and History.

The left sidebar shows a "New Worksheet" input field with a plus sign and a dropdown arrow, followed by a "Find database objects" search bar with a placeholder "Starting with..." and a "Run" button. The sidebar lists database objects:

- DEMO\_DB
  - INFORMATION\_SCHEMA
  - PUBLIC
    - No Tables or Views in this Schema
- SNOWFLAKE\_SAMPLE\_DATA
  - INFORMATION\_SCHEMA

The main workspace displays two SQL queries:

```
1 select cc_name,cc_manager from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."CALL_CENTER"
2
3
4 select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."CUSTOMER_DEMO"
```

INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWFLAKE WEB UI

The screenshot shows the Snowflake Web UI interface. On the left, there's a sidebar with database navigation. The main area is a query editor titled 'Employees' with a green checkmark icon. The code pane contains a CREATE OR REPLACE TABLE statement for 'emp' and a SELECT query. The results pane shows three rows of sample data. The history pane lists recent queries. Red numbered callouts point to various UI elements:

- 1: A red callout points to the 'EMP\_ADDR' schema entry in the sidebar.
- 2: A red callout points to the '+' button in the top right of the query editor.
- 3: A red callout points to the 'Help' link in the top right.
- 4: A red callout points to the 'Context' section in the top right, which includes roles like 'SYSADMIN' and 'PUBLIC'.
- 5: A red callout points to the 'ABC\_EMPLOYEES' schema entry in the sidebar.
- 6: A red callout points to the closing parenthesis ')' in the query code.
- 7: A red callout points to the 'SQL' tab in the results pane.
- 8: A red callout points to the 'Copy' button in the results pane.
- 9: A red callout points to the 'History' section in the bottom right.
- 10: A red callout points to the 'Columns' dropdown in the bottom right.

**INNOVATION SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

Row	ID	FIRST_NAME	LAST_NAME	CITY	POSTAL_C...
1	1	Lucas	Kidd	San Francisco	94110
2	2	May	Franklin	San Francisco	94115
3	3	Breanna	Camacho	San Francisco	94110

Status	Duration	Start	Query ID	SQL
✓	261ms	16.Feb.2018 11:18:46 AM	b81c207...	SELECT * from emp;
✓	1.34s	16.Feb.2018 11:18:44 AM	7900b58...	INSERT INTO emp (id,
✓	145ms	16.Feb.2018 11:18:44 AM	29576a6...	CREATE OR REPLACE TA
✓	141ms	16.Feb.2018 11:18:44 AM	45a046a...	use schema public;
✓	1.17s	16.Feb.2018 11:18:43 AM	97fc5781...	CREATE OR REPLACE SC
✓	1.08s	16.Feb.2018 11:18:42 AM	834a4a8...	INSERT INTO emp_ph (
✓	41ms	16.Feb.2018 11:18:41 AM	a69cb50...	INSERT INTO emp_addr

# SNOWSQL DOWNLOAD

Snowflake or contact our support team with any questions.

Partner Connect   Help   Snowsight

- [View the Snowflake Documentation >](#)
- [Visit the Community >](#)
- [Download... >](#)
- [Show help panel](#)

'."CALL\_CENTER"

INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWSQL CLIENT

## Downloads

### CLI Client (snowsql)

- JDBC Driver
- ODBC Driver
- Python Components
- Node.js Driver
- Spark Connector
- Go Snowflake Driver
- SnowCD

### CLI Client (snowsql)

Download the latest version of SnowSQL for your platform from the [Snowflake Repository](#).

For installation and configuration details, see the [Snowflake Documentation](#)



Snowflake Repository  
Linux, Windows, macOS



Snowflake GPG Public Key  
For Linux



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# SNOWSQL CLIENT

Operating System	Supported Versions
Linux	CentOS 7, 8
	Red Hat Enterprise Linux (RHEL) 7, 8
	Ubuntu 16.04, 18.04
macOS	10.14, 10.15 (Support for 10.13 was deprecated in May, 2021.)
Microsoft Windows	Microsoft Windows 8, 8.1, 10 Microsoft Windows Server 2012, 2016, 2019



**INNOVATION**  
**SOFTWARE**

LEARN. EMPOWER. INNOVATE.

# SNOWSQL CLIENT

## Install on MacOS

```
$ brew install --cask snowflake-snowsql
```

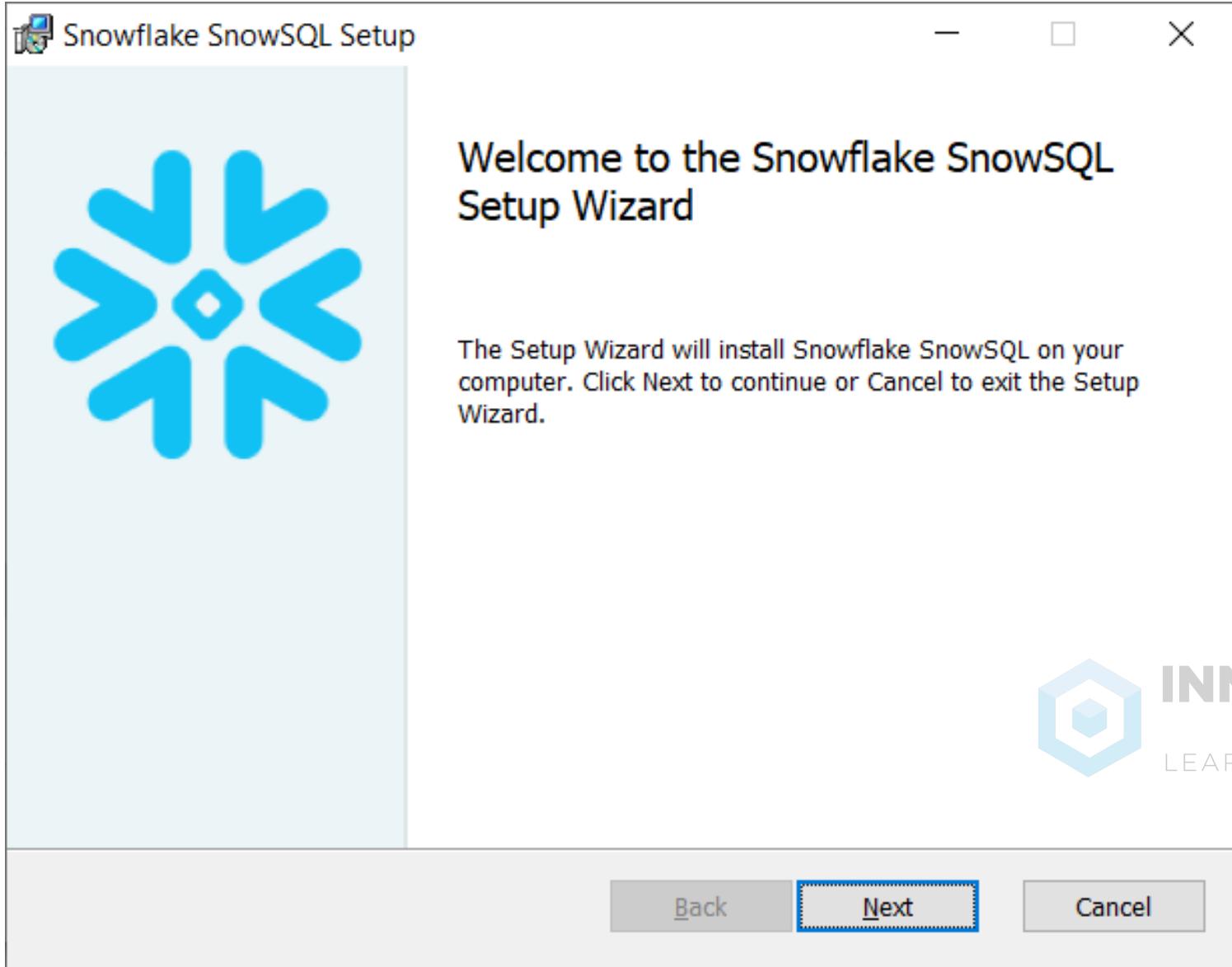
## Download the latest Windows installation

[https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/windows\\_x86\\_64/index.html](https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/windows_x86_64/index.html)



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWSQL CLIENT



 INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# SNOWSQL CONNECTION

```
| !result | !result <query id> | See the result of a query
| !set    | !set <option>=<value> | Set an option to the given value
| !source | !source <filename>, <url> | !load | Execute given sql file
| !spool  | !spool <filename>, off | Turn on or off writing results to file
| !system | !system <system command> | Run a system command in the shell
| !variables | !variables | !vars | Show all variables and their values
+-----+
+-----+
susanmartin#COMPUTE_WH@DEMO_DB.PUBLIC>!exit
Goodbye!
```

```
c:\Users\kubernetes\.snowsql>snowsql -a LBA64704.us-east-1 -u susanmartin -d snowflake_sample_data -s tpcds_sf100tcl
Password:
* SnowSQL * v1.2.20
Type SQL statements or !help
susanmartin#COMPUTE_WH@SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL>select cc_name, cc_manager from call_center CALL_CENTER
```



# SNOWFLAKE COMMUNITY



COMMUNITY

## JOIN THE SNOWFLAKE COMMUNITY!

Connect, share, and learn in the Data Cloud



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# DATA DEFINITION LANGUAGE (DDL)

- ALTER <object>
- COMMENT
- CREATE <object>
- DESCRIBE <object>
- DROP <object>
- SHOW <objects>
- USE <object>

# DATA MANIPULATION LANGUAGE (DML)

Commands for inserting, deleting, updating, and merging data in Snowflake tables:

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

# DATA MANIPULATION LANGUAGE (DML)

## Data Loading / Unloading

Commands for bulk copying data into and out of Snowflake tables:

- COPY INTO <table> (loading/importing data)
- COPY INTO <location> (unloading/exporting data)

# FILE STAGING COMMANDS

## 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  
GET  
LIST  
REMOVE

# ACCOUNT LOCATOR

The screenshot shows the Snowflake Cloud UI interface. At the top, there is a navigation bar with tabs: Worksheet, Snowflake Re, New Tab, Settings, and New Tab. Below the navigation bar is a toolbar with back, forward, and refresh buttons, along with a URL bar displaying the address lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet.

In the center, there is a message: "Enjoy your free trial! Visit our documentation to learn more about using S". Below this message is a navigation menu with icons and labels: Databases, Shares, Data Marketplace, Warehouses, Worksheets (which is selected and highlighted in grey), History, and Account.

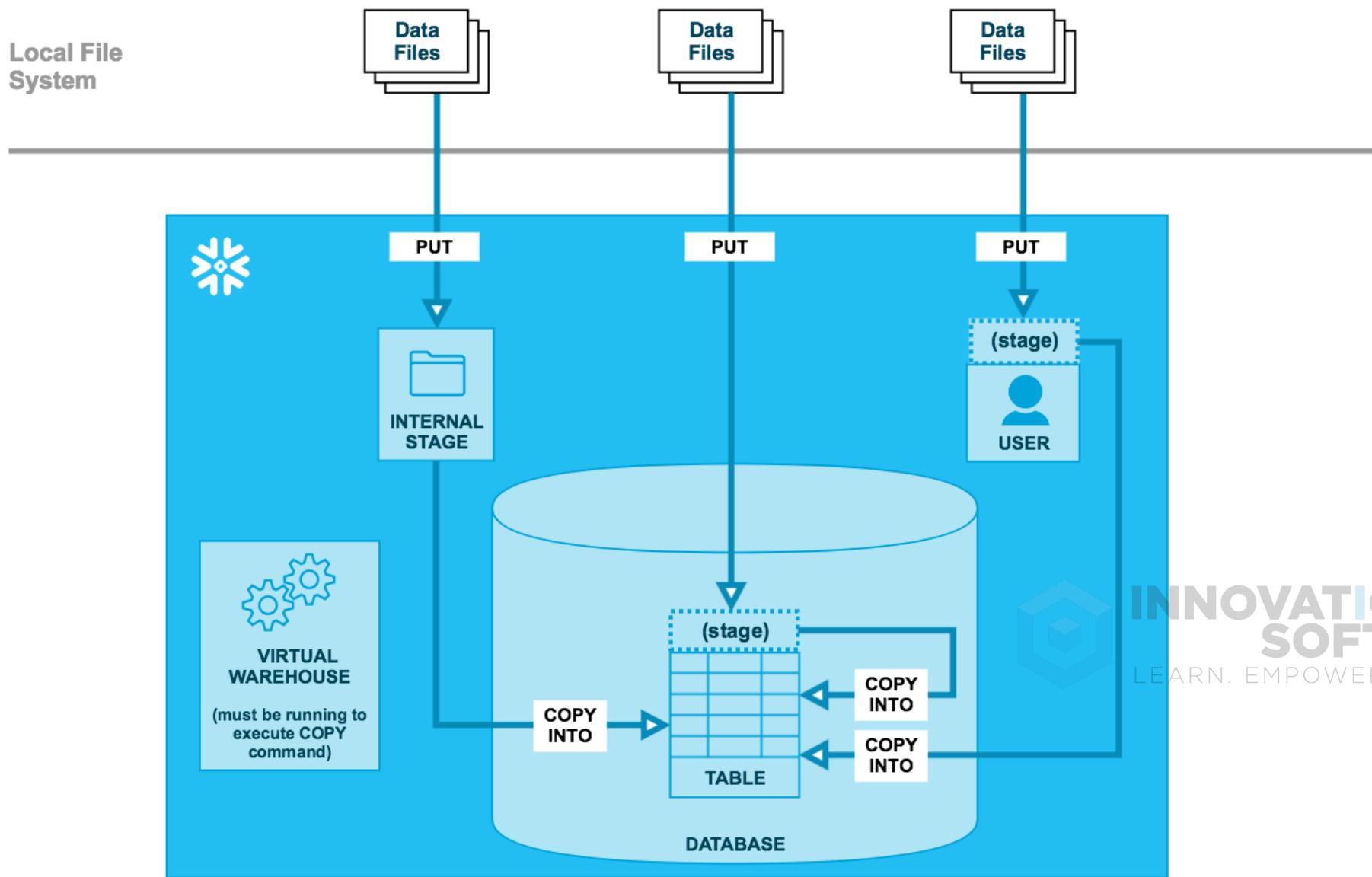
On the left side, there is a "New Worksheet" button and a search bar labeled "Find database objects" with a placeholder "Starting with...". Below the search bar, there is a list of database objects: HOUSEHOLD\_DEMOGRAPHICS and INCOME\_BAND.

On the right side, there is a query editor window. It shows a "Run" button, a checkbox for "All Queries" (unchecked), and a timestamp "Saved 1 day ago". The query itself is:

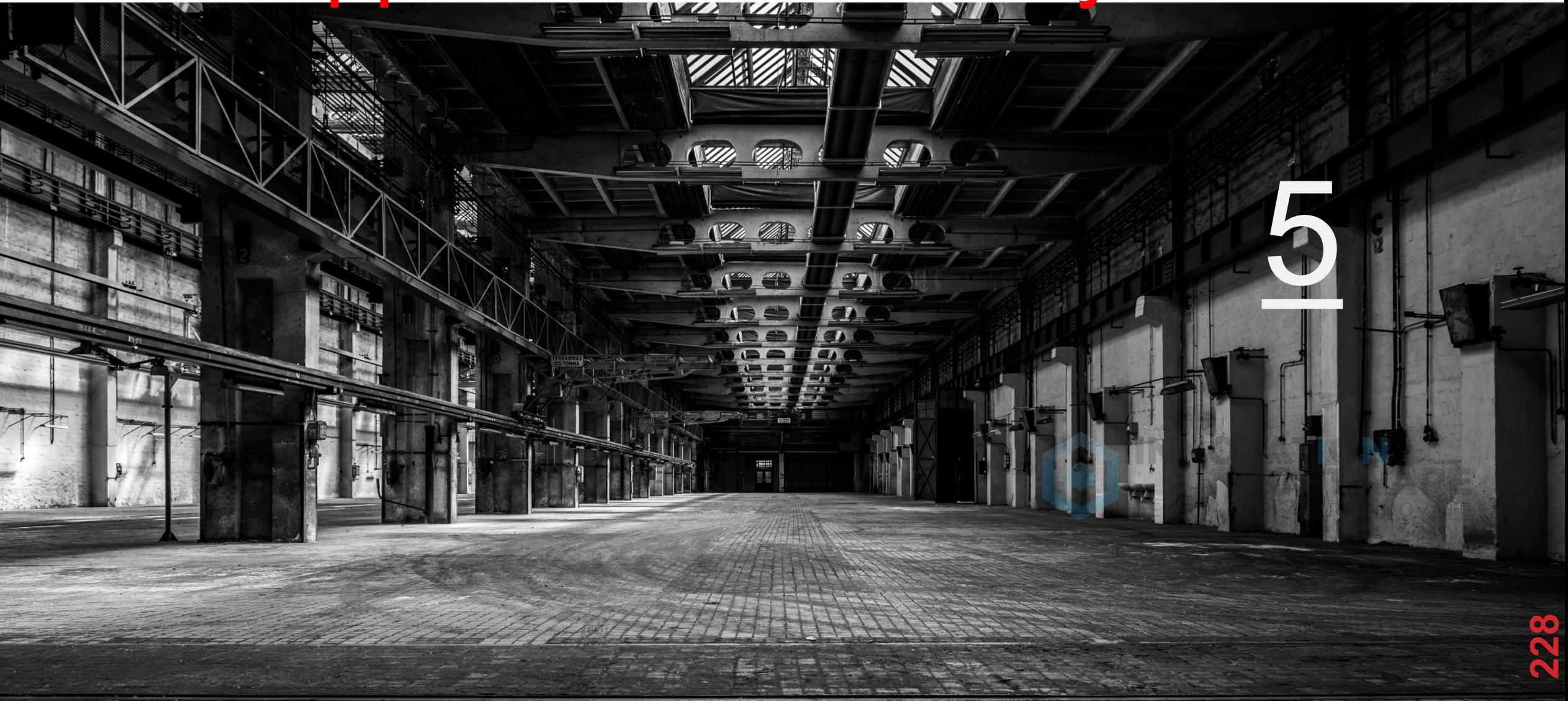
```
1 select cc_name, cc_manager from
2 "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL".
```

INNOVATION  
SOFTWARE

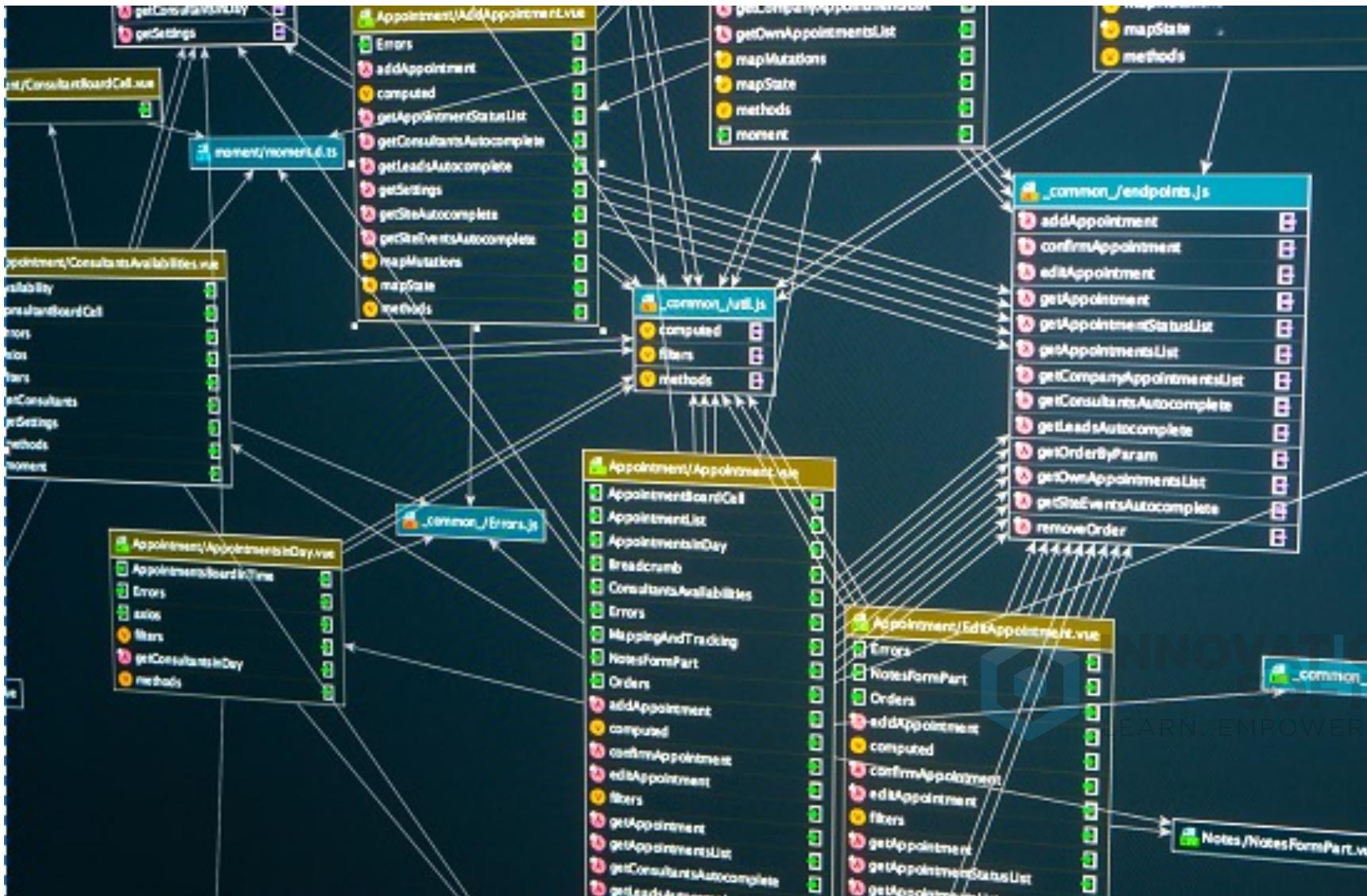
# LOCAL ONLY RESOURCES



# SQL Support for Data Analysis



# SQL SUPPORT

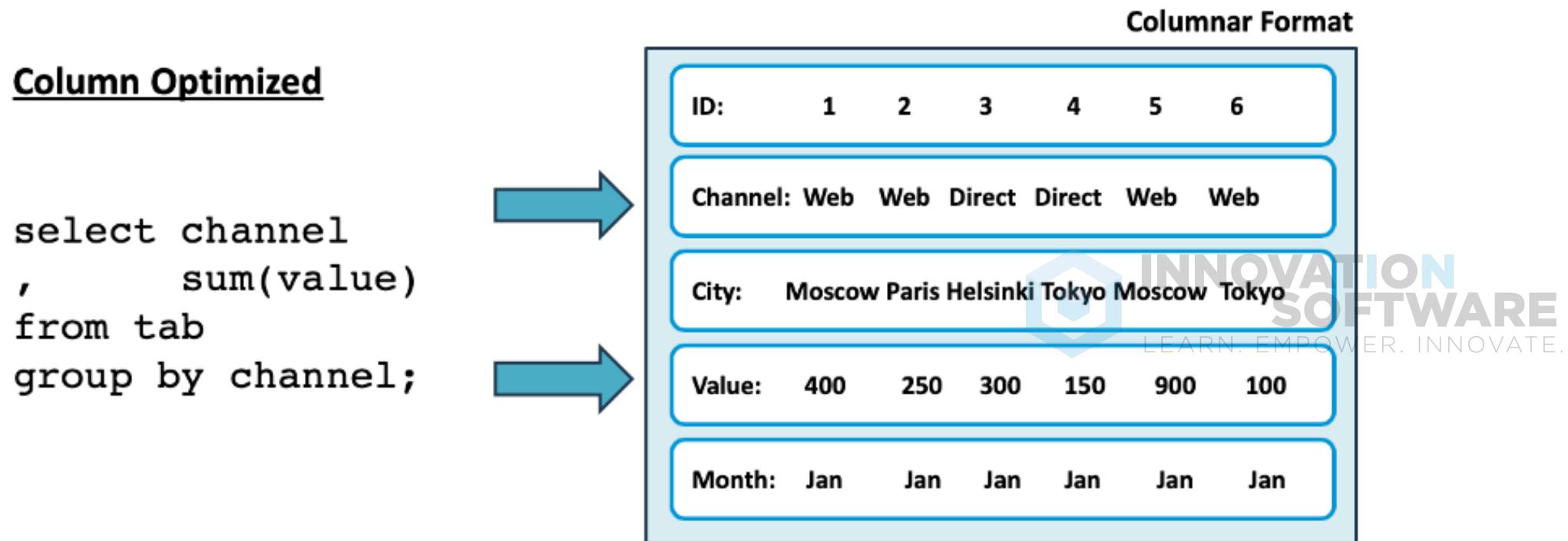


INNOVATION  
WARE  
EARN. EMPOWER. INNOVATE.

# QUERY BEST PRACTICES

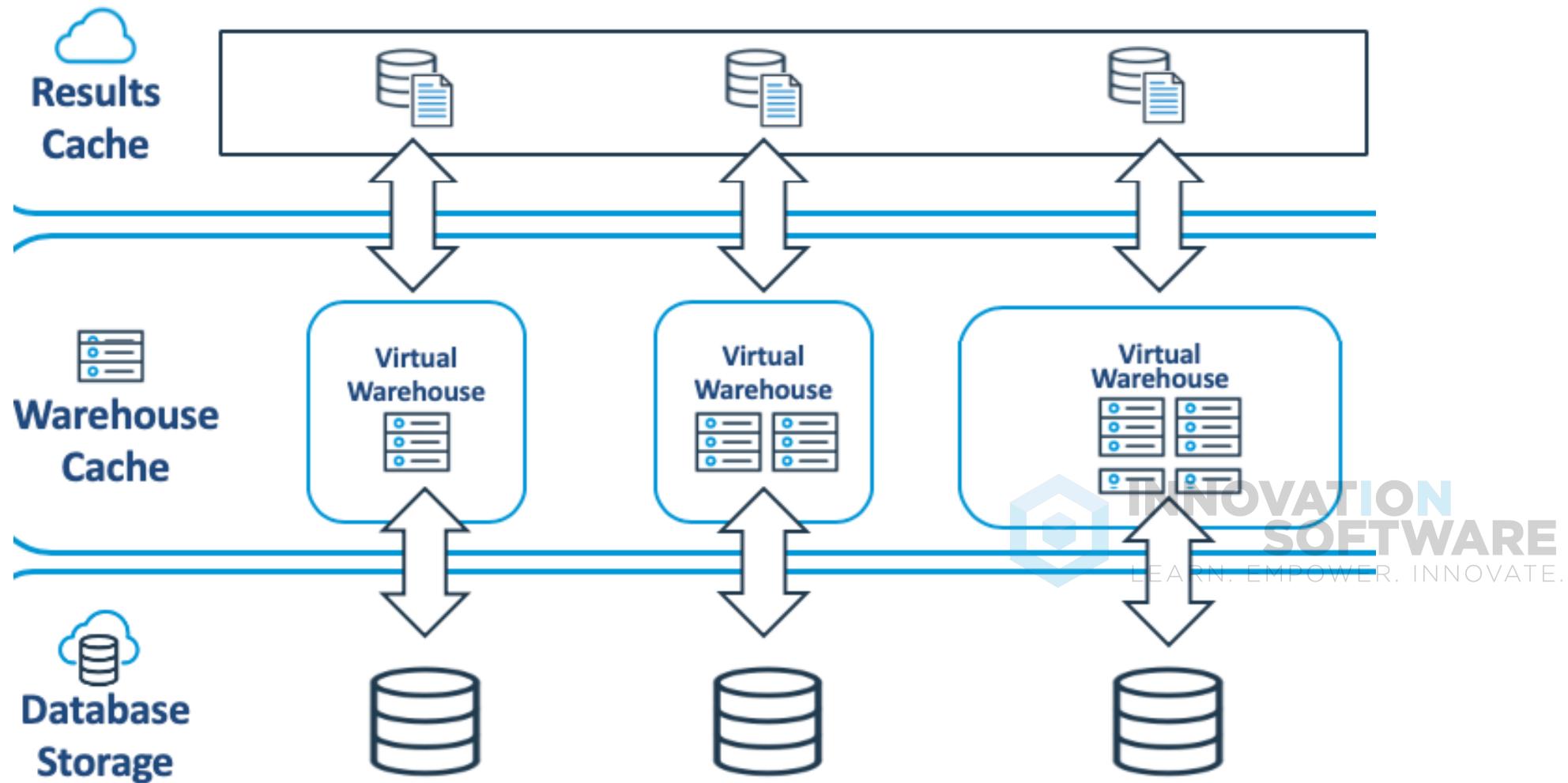
# Select the Required Columns

Like many other data analytics platforms, Snowflake uses a columnar data store which is optimized to fetch only the attributes needed for the specific query, and this is illustrated in the diagram below:



# QUERY BEST PRACTICES

## Maximize cache usage



# QUERY BEST PRACTICES

## Large query optimization

- **Table Scans:** A high value of PCT\_TABLE\_SCAN and a large number of MB\_SCANNED indicates potential poor query selectivity on large tables. Check the query WHERE clause and consider using a cluster key if appropriate.
- **Spilling:** Any value in SPILL\_TO\_LOCAL or SPILL\_TO\_REMOTE indicates a potentially large sort of operation on a small virtual warehouse. Consider moving the query to a bigger warehouse or scaling up the existing warehouse if appropriate.

# QUERY BEST PRACTICES

## Virtual Warehouse Cache Optimization

- **Fetch required attributes:** Avoid using SELECT \* in queries as this fetches all data attributes from Database Storage to the Warehouse Cache.
- **Scale Up:** While you should never scale up to tune a specific query, scaling up adds additional servers, increasing the overall warehouse cache size.
- **Consider Data Clustering:** For tables over a terabyte in size, consider creating a cluster key to maximize partition elimination. This solution both maximizes query performance for individual queries and returns fewer micro-partitions making the best use of the Warehouse Cache.

# SQL ANALYTIC FUNCTIONS

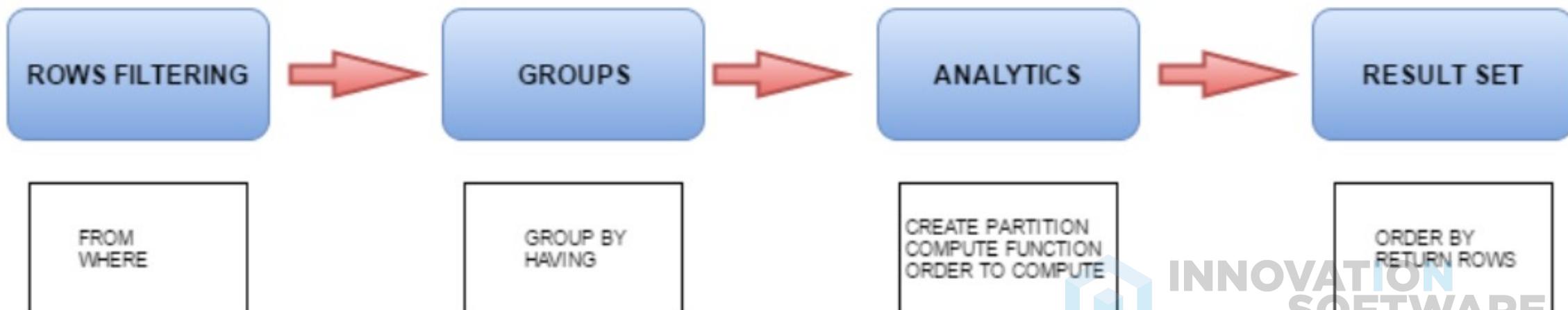
Snowflake includes sophisticated ***analytic*** and ***windowing*** functions as part of our data warehouse service. These functions use ANSI compliant **SQL**. This includes but is not limited to general aggregation functions (e.g. sum), nested virtual tables, sub queries, order by, and group by.

Functions like:

- CUME\_DIST
- DENSE\_RANK
- FIRST\_VALUE
- LAG
- LAST\_VALUE
- LEAD
- NTILE
- PERCENT\_RANK
- RANK
- ROW\_NUMBER

# ANALYTIC FUNCTIONS FLOW

Executing analytical functions organizes data into partitions, computes functions over these partitions in a specified order, and returns the result.



# HIGH PERFORMING ESTIMATION FUNCTIONS

The following Aggregate Functions are provided for using Space-Saving to estimate frequent values:

- APPROX\_TOP\_K: Returns an approximation of frequent values in the input.
- APPROX\_TOP\_K\_ACCUMULATE: Skips the final estimation step and returns the Space-Saving state at the end of an aggregation.
- APPROX\_TOP\_K\_COMBINE: Combines (i.e. merges) input states into a single output state.
- APPROX\_TOP\_K\_ESTIMATE: Computes a cardinality estimate of a Space-Saving state produced by APPROX\_TOP\_K\_ACCUMULATE and APPROX\_TOP\_K\_COMBINE.

# USER DEFINED FUNCTION (UDF)

User-defined functions (UDFs) let you extend the system to perform operations that are not available through the built-in, system-defined functions provided by Snowflake.

Snowflake currently supports the following languages for writing UDFs:

- **SQL**: A SQL UDF evaluates an arbitrary SQL expression and returns either scalar or tabular results.
- **JavaScript**: A JavaScript UDF lets you use the JavaScript programming language to manipulate data and return either scalar or tabular results.
- **Java**: A Java UDF lets you use the Java programming language to manipulate data and return either scalar or tabular results.

# OVERLOADING UDF

Snowflake supports overloading of SQL UDF names. Multiple SQL UDFs in the same schema can have the same name, as long as their argument signatures differ, either by the number of arguments or the argument types. When an overloaded UDF is called, Snowflake checks the arguments and calls the correct function.

Consider the following examples, which create two SQL UDFs named add5:

```
create or replace function add5 (n number)
    returns number
    as 'n + 5';
```

```
create or replace function add5 (s string)
    returns string
    as 's || ''5''';
```



# STORED PROCEDURE

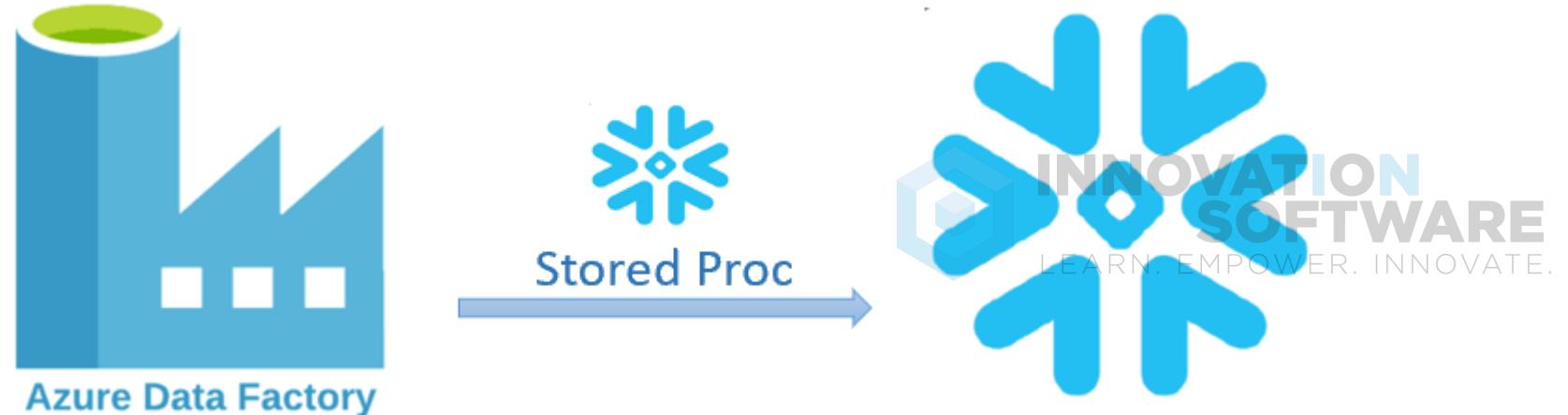
Stored procedures are loosely **similar to functions**. As with functions, a stored procedure is created once and can be executed many times. A stored procedure is created with a **CREATE PROCEDURE** command and is executed with a **CALL** command.

A stored procedure returns a single value. Although you can run **SELECT** statements inside a stored procedure, the results must be used within the stored procedure, or be narrowed to a single value to be returned.

# STORED PROCEDURE PROCESSING

Snowflake stored procedures use JavaScript and, in most cases, SQL:

- JavaScript provides the control structures (branching and looping).
- SQL is executed by calling functions in a JavaScript API.



# SNOWPARK

The Snowpark library provides an intuitive API for querying and processing data in a data pipeline.

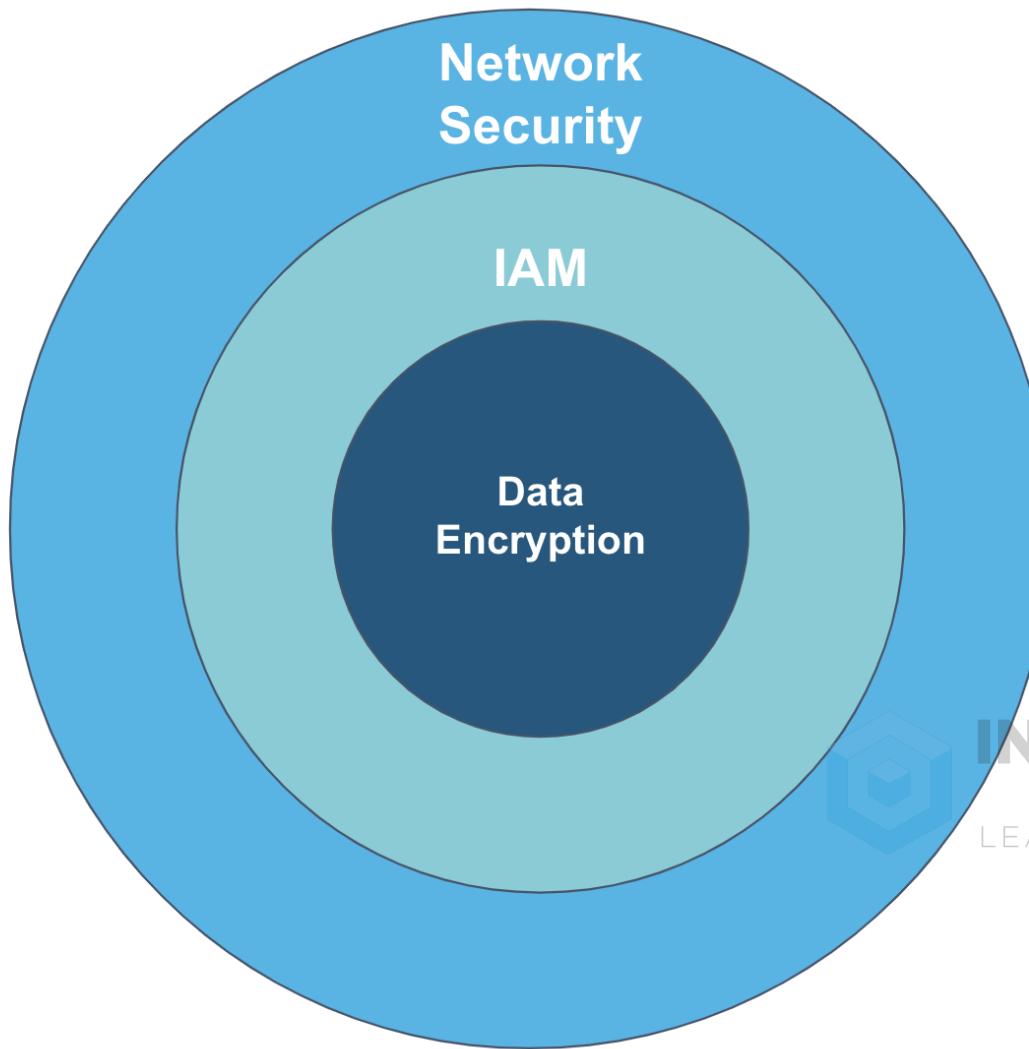
Using this library, you can build applications that process data in Snowflake without moving data to the system where your application code runs.

# Managing Security

In Snowflake



# DEFENSE IN DEPTH



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# DATA ENCRYPTION

Encrypting data means applying an encryption algorithm to translate the clear text into cipher text.

Encrypt data:

- from the time it leaves your premises
- through the Internet
- into the cloud data warehouse
- stored on disk
- moved into a staging location
- placed within a database object
- cached within a virtual data warehouse.

# AUTHENTICATION

Any data service or data warehouse should always authorize users, authenticate credentials, and grant users access only to the data they're authorized to access.

Data breaches often result from users selecting weak passwords coupled with rudimentary authentication procedures.



# IDENTITY VS ACCESS

**Identity** – who you are

**Access** – what you're allowed to do

**Authentication** – proving your identity, e.g user SUSANMARTIN with password Sn0wFl@ke

**Authorization** – the right to do something,  
SUSANMARTIN – ACCOUNTADMIN aka Snowflake  
superuser



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# AUTHORIZATION AND RBAC

**Snowflake authorization** allows you to do things in the data warehouse **based on your role**, that's **called Role-Based Access Control (RBAC)**.

# ROLE SWITCHING

Snowflake allows you to **switch roles** without doing further authentication.

The standard roles are  
ACCOUNTADMIN  
SYSADMIN  
SECURITYADMIN  
USERADMIN  
PUBLIC



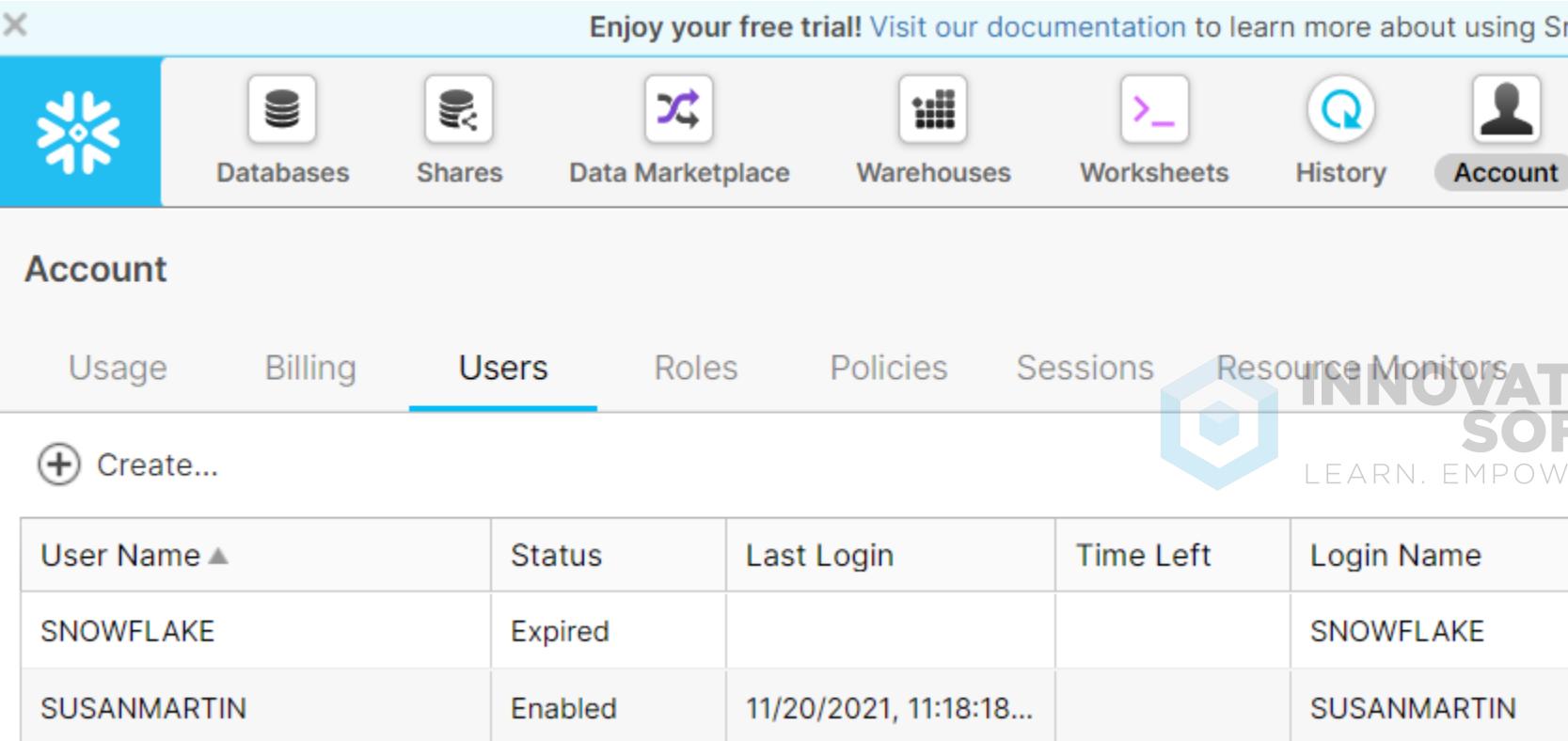
# PUBLIC DATABASE ACCESS

Snowflake provides functions to view role availability. For example, to see what roles you have available you could execute

```
select current_available_roles();
```

# USERS & ROLES

ACCOUNTADMIN and SECURITYADMIN are the only roles that have default access to the Account Tab in the Top Level Snowflake Console.



The screenshot shows the Snowflake Account console interface. At the top, there is a navigation bar with icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets, History, and Account. The 'Account' tab is currently selected. Below the navigation bar, the word 'Account' is displayed. A horizontal menu bar includes 'Usage', 'Billing', 'Users' (which is underlined in blue), 'Roles', 'Policies', 'Sessions', and 'Resource Monitors'. On the left side, there is a 'Create...' button with a plus sign icon. The main area displays a table with columns: User Name, Status, Last Login, Time Left, and Login Name. Two rows are visible: one for 'SNOWFLAKE' which is listed as 'Expired', and another for 'SUSANMARTIN' which is listed as 'Enabled' with a timestamp of '11/20/2021, 11:18:18...'. The 'INNOVATION SOFTWARE' logo is visible in the bottom right corner of the slide.

User Name ▲	Status	Last Login	Time Left	Login Name
SNOWFLAKE	Expired			SNOWFLAKE
SUSANMARTIN	Enabled	11/20/2021, 11:18:18...		SUSANMARTIN

# MANAGING USERS AND ROLES

Snowflake recommends using SCIM where supported by your Identity Provider to provision and externally manage users and roles in Snowflake.

Identity Providers can be further configured to synchronize users and roles with your Active Directory users and groups. Examples include Okta SCIM and Azure AD SCIM



# SAML OR OAUTH

SAML and OAuth are great protocols within identity management. SAML is based around user authentication and OAuth is based around authorization.

If you're using an external identity provider such as Okta, it's likely you have the ability to use both SAML and OAuth for a variety of applications and integrations. Specifically with regards to Okta, it would be recommended to use OAuth because the specification has more levels of trust and security, and you can configure different authorization flows for each one of these.

In either case, this is an area that requires experience and Subject Matter Expertise (SME) to most correctly secure.



# ACCESS CONTROL FRAMEWORK

Snowflake's approach to access control combines aspects from both of the following models:

- **Discretionary Access Control (DAC):** Each object has an owner, who can in turn grant access to that object.
- **Role-based Access Control (RBAC):** Access privileges are assigned to roles, which are in turn assigned to users.

# ACCESS CONTROL KEY CONCEPTS

**Securable object:** An entity to which access can be granted. Unless allowed by a grant, access will be denied.

**Role:** An entity to which privileges can be granted. Roles are in turn assigned to users. Note that roles can also be assigned to other roles, creating a role hierarchy.

**Privilege:** A defined level of access to an object. Multiple distinct privileges may be used to control the granularity of access granted.

**User:** A user identity recognized by Snowflake, whether associated with a person or program.

# SNOWFLAKE AUTHENTICATION METHODS

## Snowflake Drivers, CLI

Password

OAuth

Key Pair

External Browser

Okta native auth

## Snowflake UI

Password

SAML

## Snowpipe

Key Pair



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# AUTHENTICATION BEST PRACTICE

**Preference #1:** OAuth (either Snowflake OAuth or External OAuth)

**Preference #2:** External Browser, if it's a desktop application that doesn't support OAuth

**Preference #3:** Okta native authentication, if you're using Okta, and the app supports this method while not supporting OAuth or external browser authentication yet.

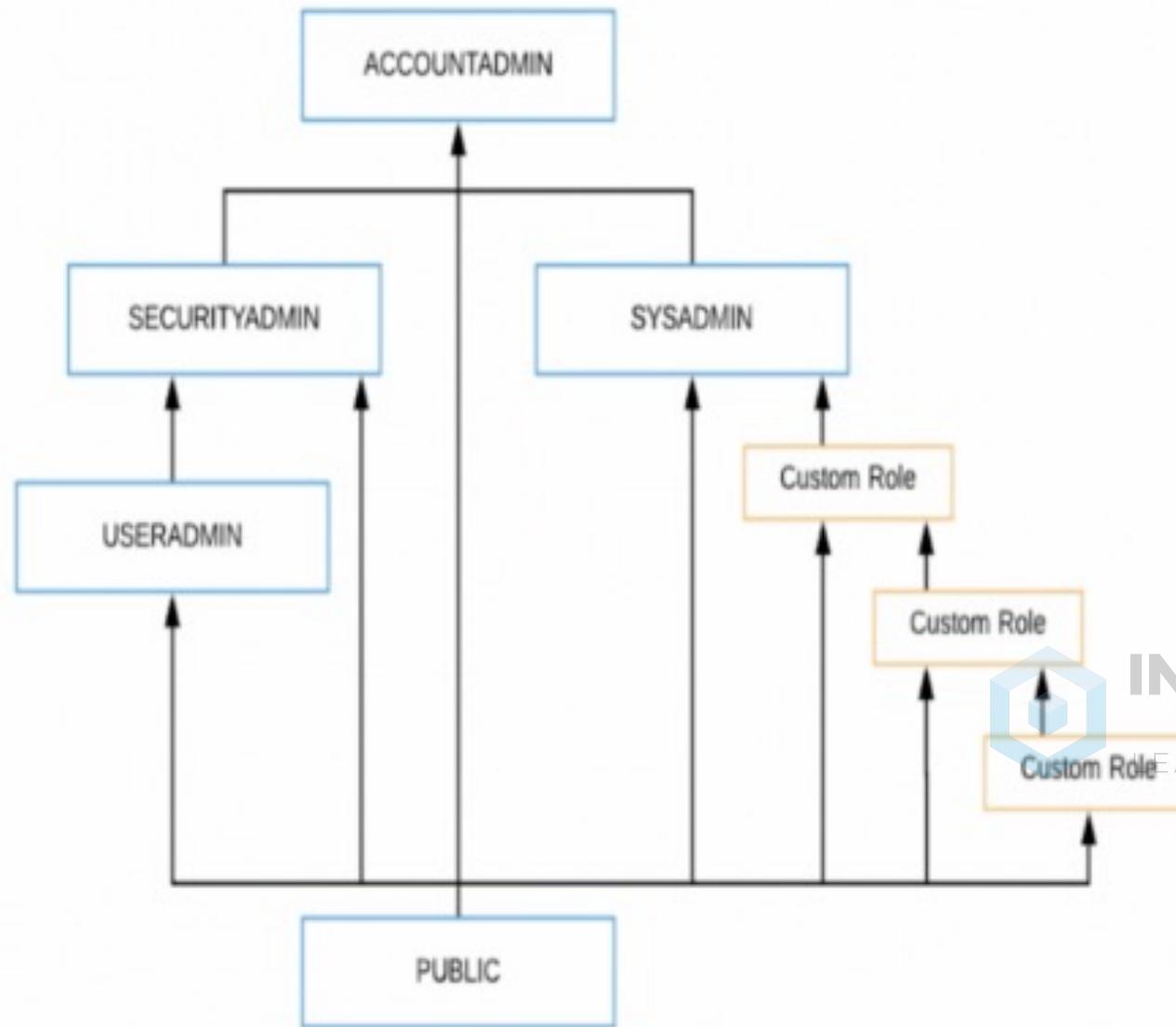
**Preference #4:** Key Pair Authentication, mostly used for service account users. Since this requires the client application to manage private keys, complement it with your internal key management software.

**Preference #5:** Password, this should be the last option for applications that don't support any of the above options. This option is commonly used for service account users connecting from 3rd party ETL apps.



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# ROLE-BASED ACCESS CONTROL (RBAC)



# LOGICAL RBAC LEVELS

**Access Roles:** These roles will be the lowest level which will have actual access privileges on DB objects. Maintain distinct access roles based on user requirements at a schema level.

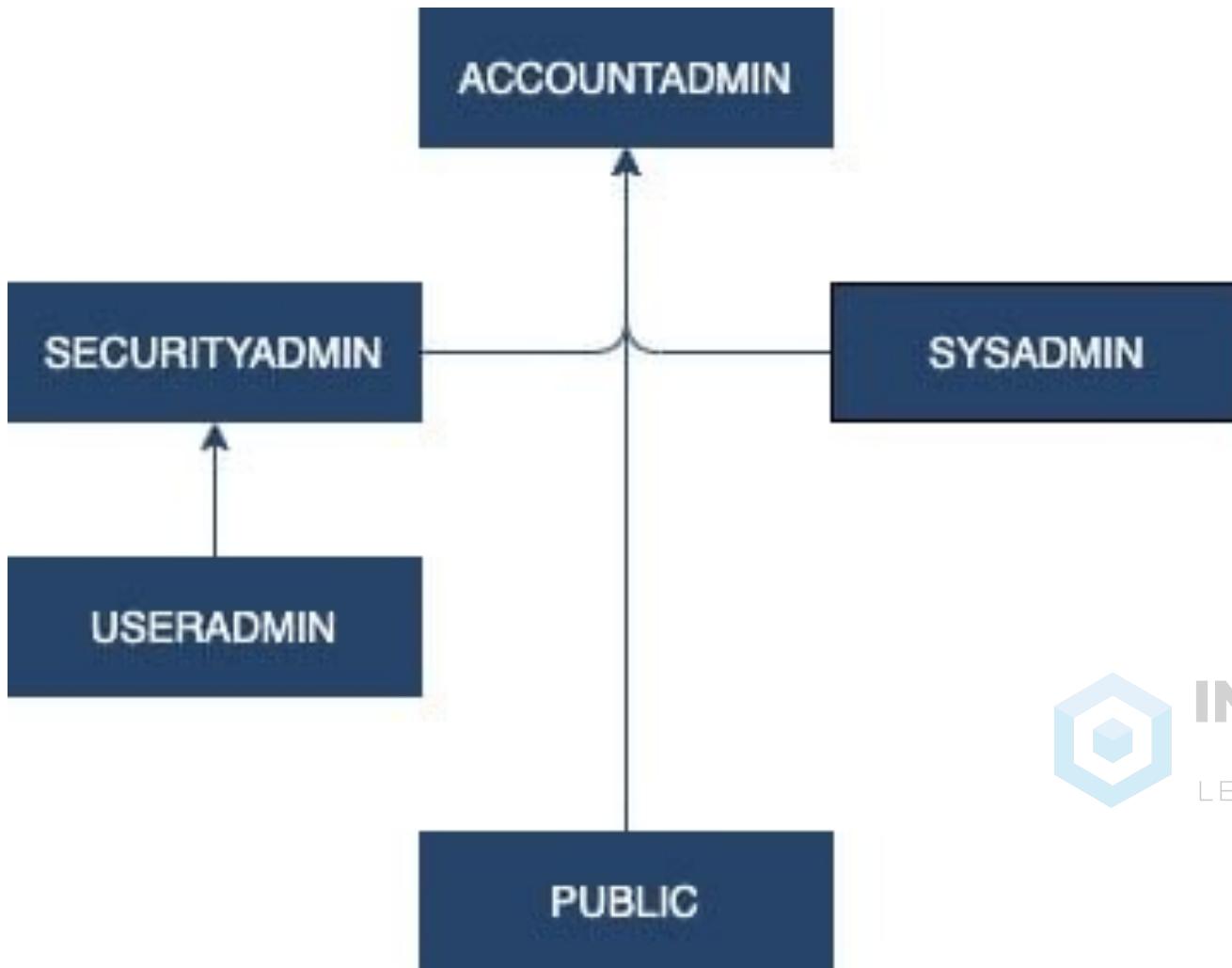
**Functional Roles:** These roles map to the actual real-world roles of the users in the organization and are assigned to the Snowflake users.

**Domain Roles:** If the organization has a requirement to have multiple independent domains under the same account, this will help to realize that. E.g. when there needs to be a separation of production or UAT/development domains.



**System Roles:** Native Snowflake system role to which all the Domain roles should be rolling up to.

# SYSTEM-DEFINED ROLES



# SYSTEM-DEFINED ROLES

## ACCOUNTADMIN

This is the highest level role in Snowflake. It should be heavily limited in access and encapsulates SYSADMIN and SECURITYADMIN

## SECURITYADMIN

This role is used for managing any object grant globally, and therefore is granted the MANAGE GRANTS privilege by default. This role also inherits the USERADMIN role.

## USERADMIN

This role is dedicated to user/role management. By default, this role can create/modify users and their respective roles (assuming those roles/users haven't been transferred to another role).

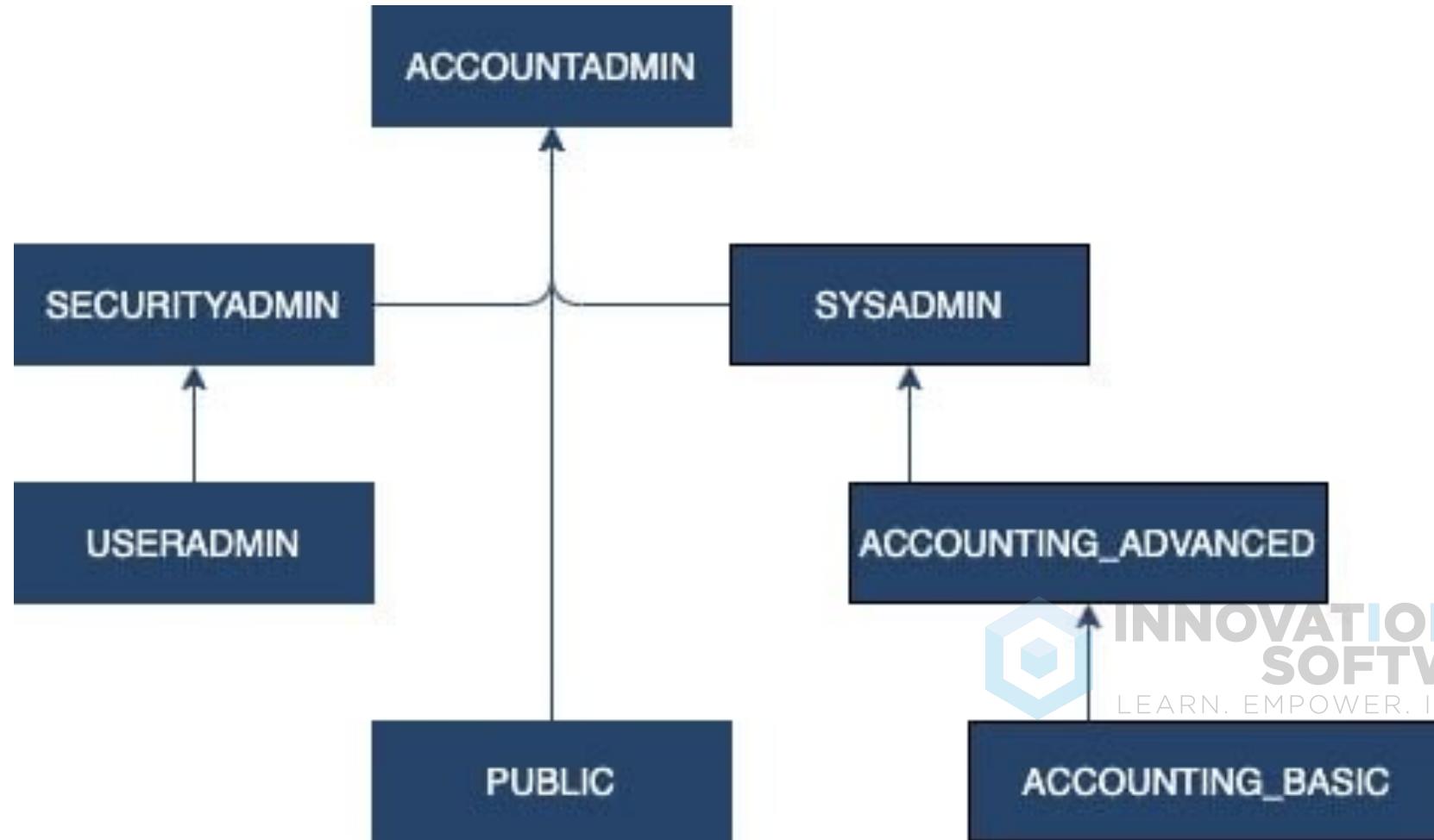
## SYSADMIN

This role is dedicated to system object management. It is recommended by Snowflake to assign all custom roles to the SYSADMIN role, so this role can grant these privileges to other roles.

## PUBLIC

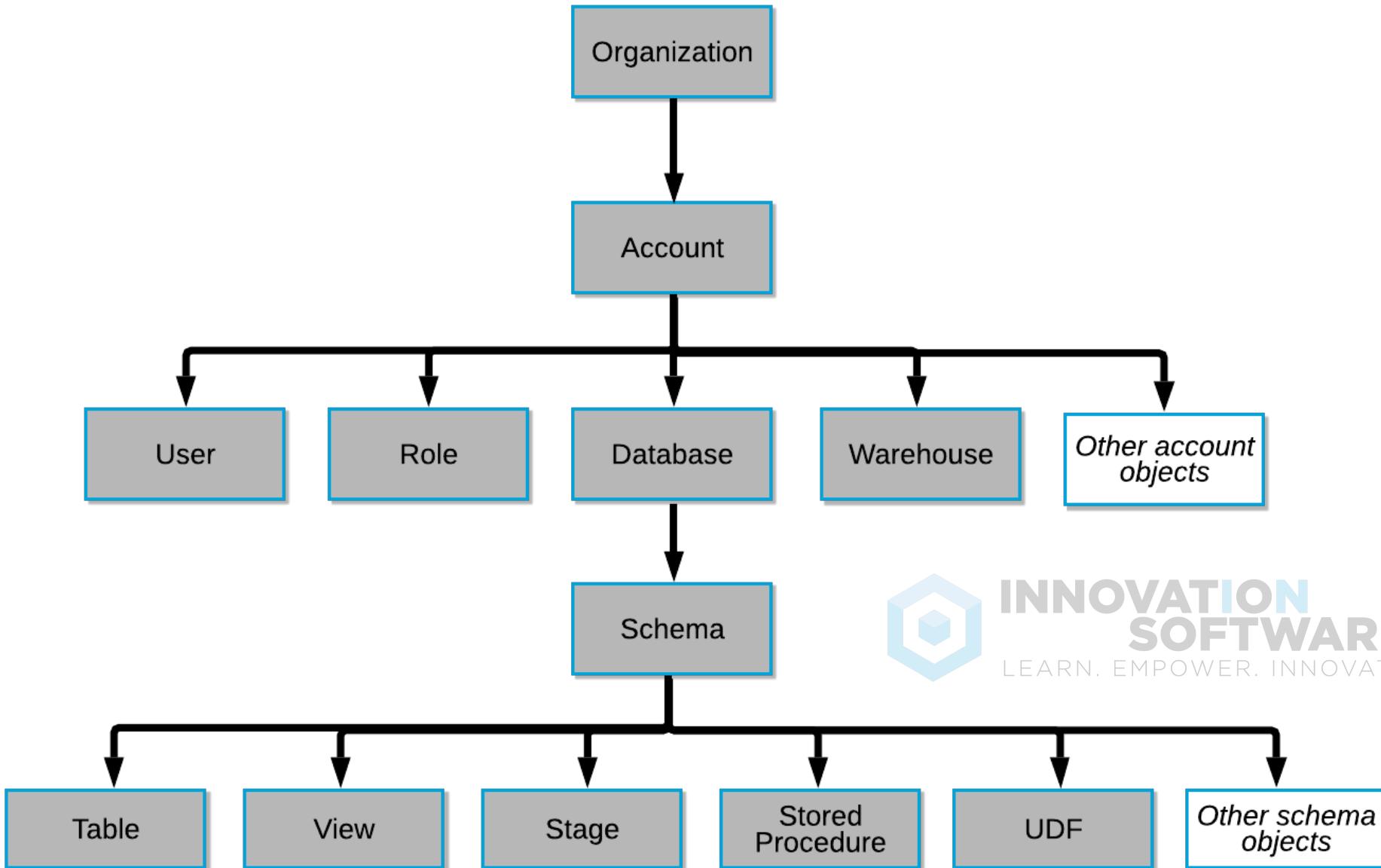
This is a pseudo role that every user/role in the account gets. Things owned by the public role are, well, public!

# CUSTOM ROLES



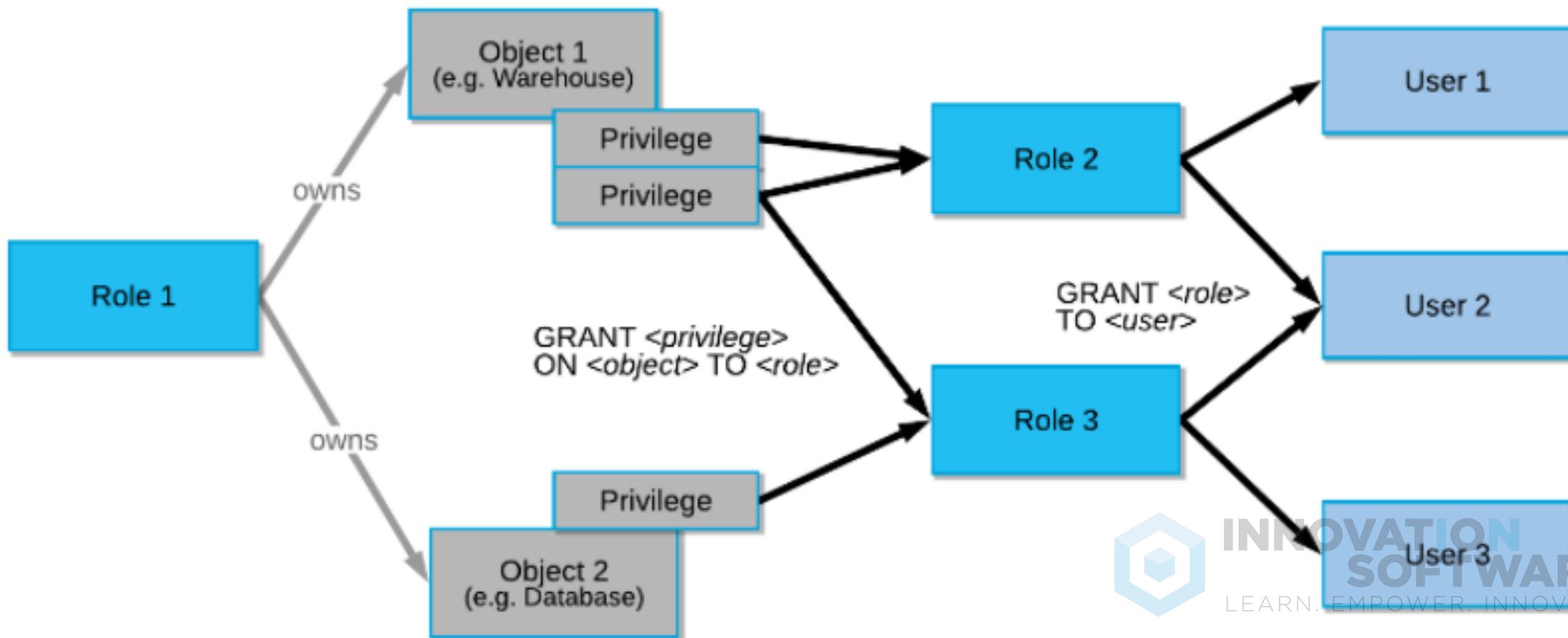
**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# SECURABLE OBJECTS



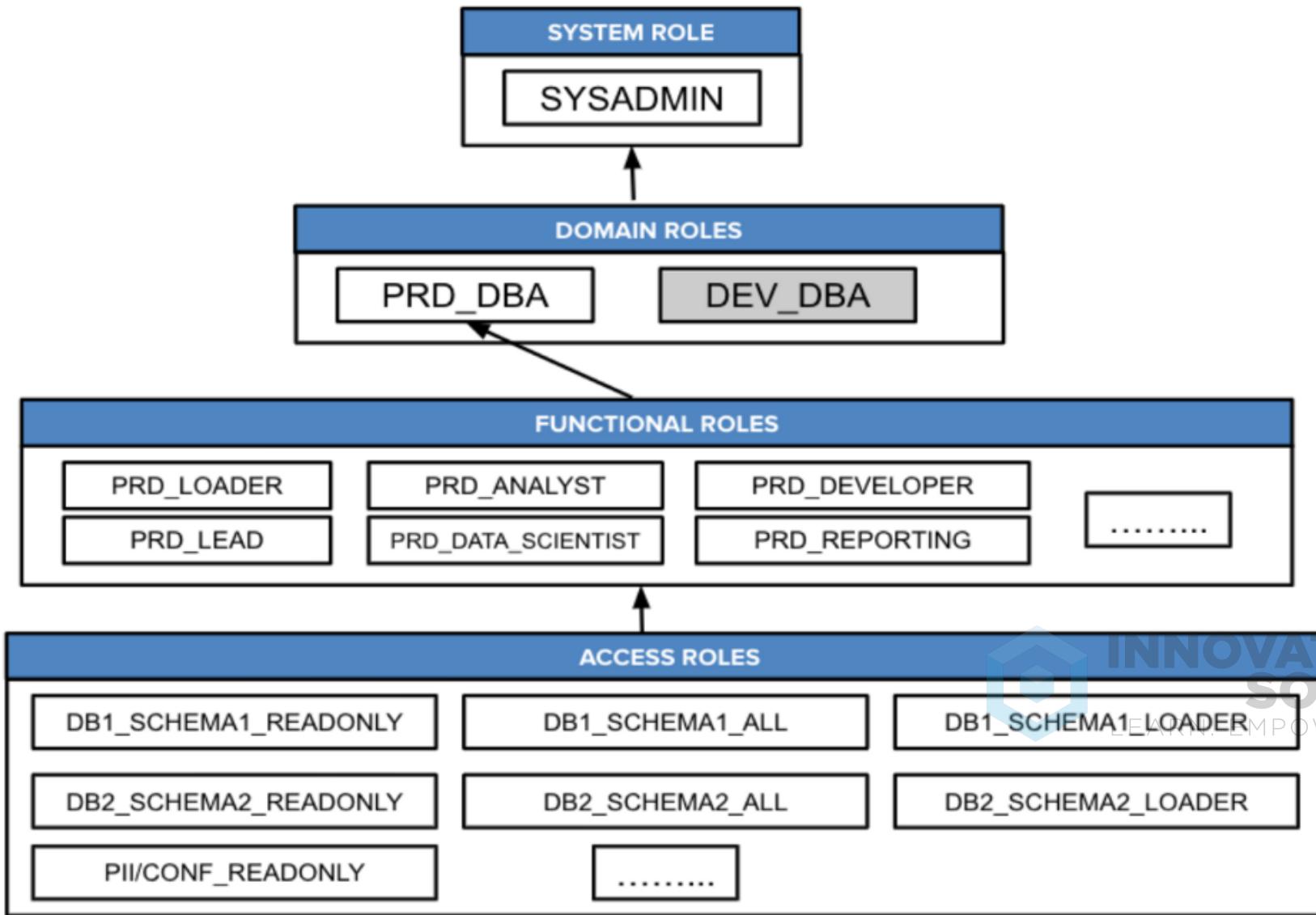
**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# PRIVILEGES

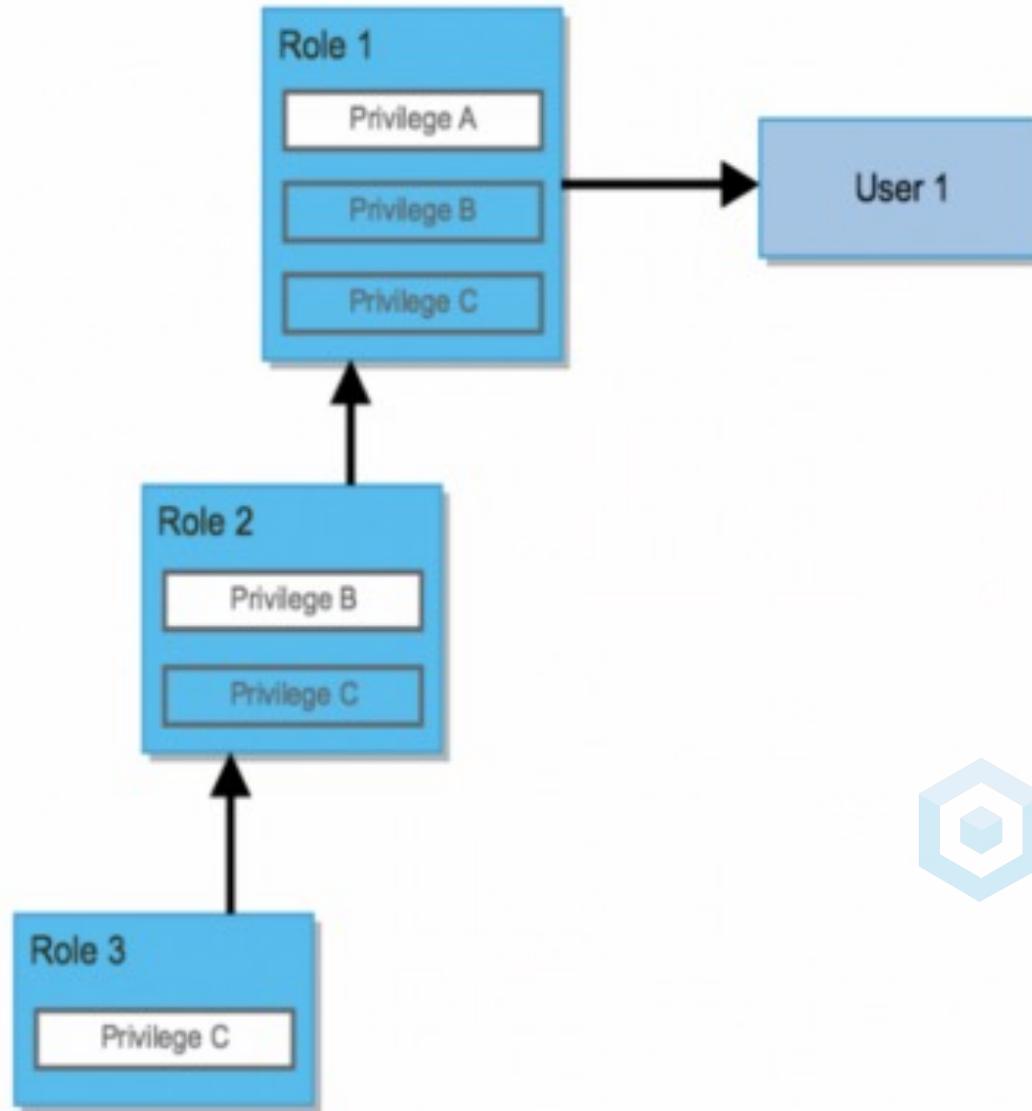


INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# RBAC ROLE HIERARCHY



# PRIVILEGE INHERITANCE



# DEFAULT TRIAL OWNER ROLE

Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact support if you have any questions.

Databases Shares Data Marketplace Warehouses Worksheets History Account

Account

Usage Billing Users Roles Policies Sessions Resource Monitors Reader Accounts

+ Create... View Type User Preferences

User Name ▲	Status	Login Name	Display Name	Default Warehouse	Default Namesp
GEOFFSNOW55113	Enabled	GEOFFSNOW55...	GEOFFSNOW55...	COMPUTE_WH	
SNOWFLAKE	Expired	SNOWFLAKE	SNOWFLAKE		GEOFFSNOW55113

Disable User Reset Password Drop

Login Name: GEOFFSNOW55113  
Display Name: GEOFFSNOW55113  
Default Role: ACCOUNTADMIN  
Default Warehouse: COMPUTE\_WH  
Default Namespace: SNOWFLAKE  
Last Login: Today at 3:07:12 AM  
Status: Enabled

MFA



INNOVATION SOFTWARE  
BUILD. LEARN. EMPOWER. INNOVATE.

# EXTERNAL VALIDATION

Snowflake allows the using of external systems to validate user access using Oauth 2.0 for SSO experience. These services include:

Snowflake supports the following external authorization servers, custom clients, and partner applications:

- [Okta](#)
- [Microsoft Azure AD](#)
- [Ping Identity PingFederate](#)
- [External OAuth Custom Clients](#)
- [Microsoft Power BI](#)
- [Sigma](#)

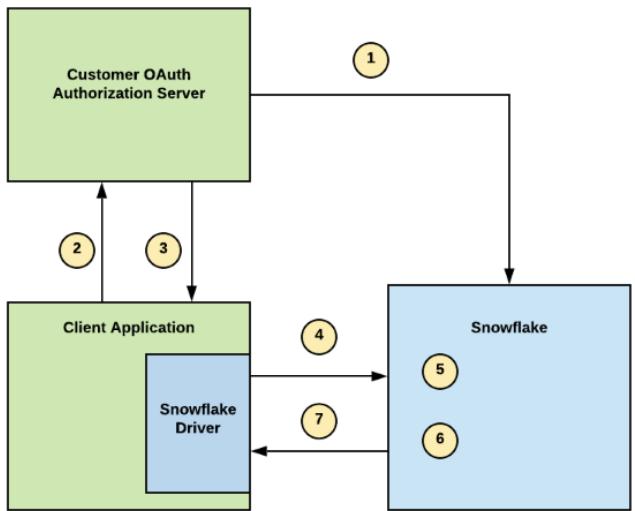


# EXTERNAL VALIDATION

## Use Cases & Benefits

- 1.Snowflake delegates the token issuance to a dedicated authorization server to ensure that the OAuth Client and user properly authenticate
- 2.Customers can integrate their policies for authentication (e.g. multi-factor, subnet, biometric) and authorization (e.g. no approval, manager approval required) into the authorization server.
- 3.For programmatic clients that can access Snowflake and users that only initiate their Snowflake sessions through External OAuth, no additional authentication configuration (i.e. set a password) is necessary in Snowflake
- 4.Clients can authenticate to Snowflake without browser access, allowing ease of integration with the External OAuth server.
- 5.Snowflake's integration with External OAuth servers is cloud-agnostic.

# EXTERNAL VALIDATION



1. Configure your External OAuth authorization server in your environment and the security integration in Snowflake to establish a trust.
2. A user attempts to access Snowflake data through their business intelligence application, and the application attempts to verify the user.
3. On verification, the authorization server sends a JSON Web Token (i.e. OAuth token) to the client application.
4. The Snowflake driver passes a connection string to Snowflake with the OAuth token.
5. Snowflake validates the OAuth token.
6. Snowflake performs a user lookup.
7. On verification, Snowflake instantiates a session for the user to access data in Snowflake based on their role.



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Managing Security



Role switching requires you to log out and log back into the Snowflake console:

- A: True
- B: False

# POP QUIZ:

## Managing Security



Role switching requires you to log out and log back into the Snowflake console:

A: True

B: False

# POP QUIZ:

## Managing Security



Choose the Snowflake Securable Objects

- A: Schema
- B: JSON
- C: Database
- D: AWS S3
- E: Views



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Managing Security



Choose the Snowflake Securable Objects

- A: Schema
- B: JSON
- C: Database
- D: AWS S3
- E: Views



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Managing Security



Snowflake access includes

- A: Visibility of databases
- B: Ability to do data sharing
- C: Logging in with your password
- D: RBAC
- E: Authentication



# POP QUIZ:

## Managing Security



Snowflake access control includes

A: Visibility of databases

B: Ability to do data sharing

C: Logging in with your password

D: RBAC

E: Authentication



# POP QUIZ:

## Managing Security



Default security roles include:

- A: SECURITYADMIN
- B: DBA
- C: ADMIN
- D: PUBLIC
- E: USEADMIN



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Managing Security



Default security roles include (select all that apply):

- A: SECURITYADMIN
- B: DBA
- C: ADMIN
- D: PUBLIC
- E: USEADMIN



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## Managing Security



Which role is the Snowflake super user equivalent?



# POP QUIZ:

## Managing Security



Which role is the Snowflake super user equivalent?

**ACCOUNTADMIN**



# POP QUIZ:

## Managing Security



Which privileges are automatically assigned to SYSADMIN

- A: CREATE DATABASE
- B: MANAGE GRANTS
- C: CREATE WAREHOUSE
- D: All of the Above



# POP QUIZ:

## Managing Security



Which privileges are automatically assigned to SYSADMIN

- A: CREATE DATABASE
- B: MANAGE GRANTS
- C: CREATE WAREHOUSE
- D: All of the Above



# POP QUIZ:

## Managing Security



What is the Snowflake chaining of roles for a user called?



# POP QUIZ:

## Managing Security



What is the Snowflake chaining of roles for a user called?

Privilege Inheritance



# POP QUIZ:

## Managing Security



When you sign up for a trial you are directly awarded which RBAC role

- A: SECURITYADMIN
- B: OWNER
- C: JEDIMASTER
- D: USERADMIN
- E: ACCOUNTADMIN

# POP QUIZ:

## Managing Security



When you sign up for a trial you are directly awarded which RBAC role

- A: SECURITYADMIN
- B: OWNER
- C: JEDIMASTER
- D: USERADMIN
- E: ACCOUNTADMIN

# POP QUIZ:

## Managing Security



What are the roles your default ACCOUNTADMIN inherit when you create your Snowflake account?



# POP QUIZ:

## Managing Security



What are the roles your default ACCOUNTADMIN inherit when you create your Snowflake account?

USERADMIN

SYSADMIN

SECURITYADMIN



# POP QUIZ:

## Managing Security



Which statements are true about our Trial Account User

- A: User defaults to PUBLIC role
- B: User defaults to SYSADMIN role
- C: User cannot switch their role
- D: User can switch to any other available role
- E: If user logs out and in their default role changes to ACCOUNTADMIN

# POP QUIZ:

## Managing Security



Which statements are true about our Trial Account User

- A: User defaults to PUBLIC role
- B: User defaults to SYSADMIN role
- C: User cannot switch their role
- D: User can switch to any other available role
- E: If user logs out and in their default role changes to ACCOUNTADMIN

**INNOVATION IN SOFTWARE**  
LEARN. LEAP. POWER. INNOVATE.

# Data Modeling



# SCHEMA ON READ VS SCHEMA ON WRITE

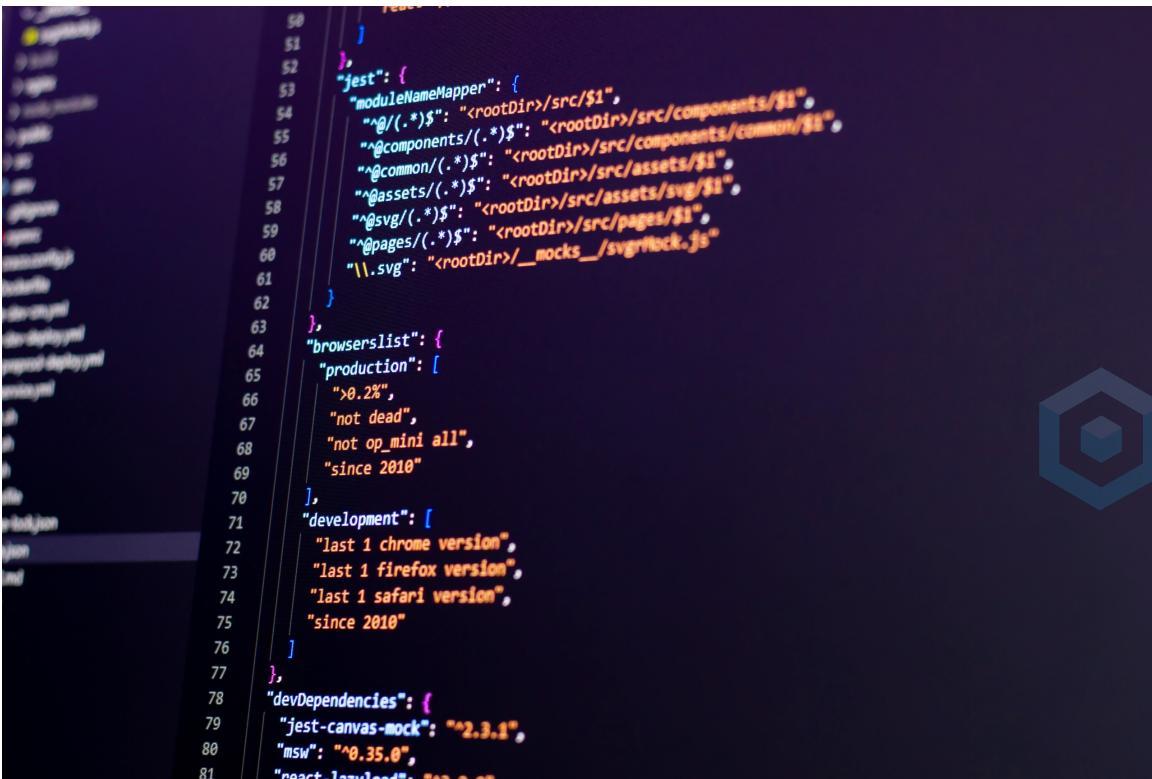
Schema on Read is when the schema is automatically inferred from the data sources

Schema on Write is when you must write the schema

	Schema on Write	Schema on Read
Schema	User has to define a schema	Schema is inferred from the data
Data	Structured and relational	Unstructured and Structured
End User Experience	The only queryable data is pre-selected	Allows richer data exploration
Positive Features	Lightweight	Adaptable

# TRANSFORMING JSON TO RDBMS

Load JSON files into Snowflake can be done either Schema on Read or Schema on Write. JSON files are common data files that allow data to be moved between numerous systems, regardless of system and schema



```
50
51
52 },
53 "jest": {
54   "moduleNameMapper": {
55     "@/(.*)$": "<rootDir>/src/$1",
56     "@components/(.*)$": "<rootDir>/src/components/common/$1",
57     "@common/(.*)$": "<rootDir>/src/components/common/$1",
58     "@assets/(.*)$": "<rootDir>/src/assets/svg/$1",
59     "@svg/(.*)$": "<rootDir>/src/assets/svg/$1",
60     "@pages/(.*)$": "<rootDir>/src/pages/$1",
61     "\\.svg": "<rootDir>/__mocks__/svg$!lock.js"
62   }
63 },
64 "browserslist": {
65   "production": [
66     ">0.2%",
67     "not dead",
68     "not op_mini all",
69     "since 2010"
70   ],
71   "development": [
72     "last 1 chrome version",
73     "last 1 firefox version",
74     "last 1 safari version",
75     "since 2010"
76   ]
77 },
78 "devDependencies": {
79   "jest-canvas-mock": "2.3.1",
80   "msw": "0.35.0",
81   "react-lazyload": "1.3.2"
82 }
```

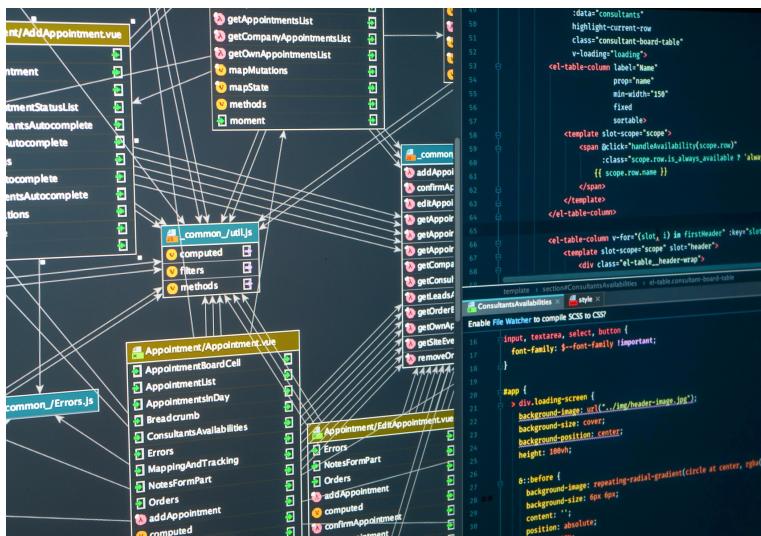


INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO RDBMS

Load JSON files into Snowflake can be done either Schema on Read (JSON Column) or Schema on Write (Predefined table). JSON files are common data files that allow data to be moved between numerous systems, regardless of system and schema

For this section, we will need to use SnowSQL



# INNOVATION SOFTWARE

LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO RDBMS

For example,  
we can  
generate a  
sales table  
using the  
following JSON

```
[{"location": {"state_city": "MA-Lexington", "zip": "40503"}, "sale_date": "2017-3-5", "price": "275836"}, {"location": {"state_city": "MA-Belmont", "zip": "02478"}, "sale_date": "2017-3-17", "price": "392567"}, {"location": {"state_city": "MA-Winchester", "zip": "01890"}, "sale_date": "2017-3-21", "price": "389921"}]
```

# TRANSFORMING JSON TO RDBMS

On the left, we are pre-defining the table structure while we will load everything into a JSON column on the right

```
USE ROLE ACCOUNTADMIN;

CREATE OR REPLACE DATABASE SNOWTEST;
CREATE OR REPLACE SCHEMA SNOWTEST.PUBLIC;

CREATE OR REPLACE TEMPORARY TABLE SNOWTEST.PUBLIC.home_sales (
    city STRING,
    zip STRING,
    state STRING,
    type STRING DEFAULT 'Residential',
    sale_date timestamp_ntz,
    price STRING
);

CREATE OR REPLACE FILE FORMAT sf_tut_json_format
TYPE = JSON;

CREATE OR REPLACE TEMPORARY STAGE sf_tut_stage
FILE_FORMAT = sf_tut_json_format;
```

```
USE ROLE ACCOUNTADMIN;

CREATE OR REPLACE DATABASE SNOWTEST;
CREATE OR REPLACE SCHEMA SNOWTEST.PUBLIC;

CREATE OR REPLACE TABLE home_sales (
    json_column variant
);

CREATE OR REPLACE FILE FORMAT sf_tut_json_format
TYPE = JSON;
```

```
CREATE OR REPLACE TEMPORARY STAGE sf_tut_stage
FILE_FORMAT = sf_tut_json_format;
```



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO RDBMS

JSON  
Column

VS  
  
Pre-  
defined  
table

```
PUT file://C:\Users\kwame\Downloads\sales.json @sf_tut_stage AUTO_COMPRESS=TRUE;  
PUT file://C:/Users/kwame/Downloads/sales.json @sf_tut_stage AUTO_COMPRESS=TRUE;  
  
COPY INTO home_sales(json_column)  
    FROM (SELECT *  
          FROM @sf_tut_stage/sales.json.gz t)  
    ON_ERROR = 'continue';  
  
SELECT * FROM home_sales;
```

```
PUT file://C:\Users\kwame\Downloads\sales.json @sf_tut_stage AUTO_COMPRESS=TRUE; -- Windows  
PUT file://C:/Users/kwame/Downloads/sales.json @sf_tut_stage AUTO_COMPRESS=TRUE; -- Linux/MacOS
```

```
COPY INTO home_sales(city, state, zip, sale_date, price)  
    FROM (SELECT SUBSTR($1:location.state_city,4),  
          SUBSTR($1:location.state_city,1,2),  
          $1:location.zip,  
          to_timestamp_ntz($1:sale_date),  
          $1:price  
    FROM @sf_tut_stage/sales.json.gz t)  
    ON_ERROR = 'continue';  
  
SELECT * FROM SNOWTEST.PUBLIC.home_sales;
```



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO RDBMS

## JSON Column vs Pre-defined Table

```
KEVERETTEXIT#COMPUTE_WH@SNOWTEST.PUBLIC>SELECT * FROM home_sales;  
+-----+  
| JSON_COLUMN  
+-----+  
| {  
|   "location": {  
|     "state_city": "MA-Lexington",  
|     "zip": "40503"  
|   },  
|   "price": "275836",  
|   "sale_date": "2017-3-5"  
| }  
| {  
|   "location": {  
|     "state_city": "MA-Belmont",  
|     "zip": "02478"  
|   },  
|   "price": "392567",  
|   "sale_date": "2017-3-17"  
| }  
| {  
|   "location": {  
|     "state_city": "MA-Winchester",  
|     "zip": "01890"  
|   },  
|   "price": "389921",  
|   "sale_date": "2017-3-21"  
| }  
+-----+
```

CITY	ZIP	STATE	TYPE	SALE_DATE	PRICE
Lexington	40503	MA	Residential	2017-03-05 00:00:00.000	275836
Belmont	02478	MA	Residential	2017-03-17 00:00:00.000	392567
Winchester	01890	MA	Residential	2017-03-21 00:00:00.000	389921

3 Row(s) produced. Time Elapsed: 0.228s



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO 3NF

Third normal form (3NF) is a database schema design approach for relational databases which uses normalizing principles to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management.

When loading JSON files, manipulate the JSON to implement 3NF design principles.

# TRANSFORMING JSON TO 3NF

For this example, we want to convert the following JSON structure to 3NF.

This file including numerous embedded arrays that need to be separated into another table (i.e. course)

```
student: {  
    "ID": 1,  
    "Name": "Jill",  
    "Age": 25  
},  
location: [  
    {  
        "name": "MI",  
        "active": "Y"  
    },  
    {  
        "name": "TX",  
        "active": "N"  
    }  
,  
courses: [  
    "Math",  
    "Science",  
    "English",  
    "History"  
]
```

# TRANSFORMING JSON TO 3NF

We can select fields within the JSON using the following syntax:

<json column>:<level 1>,<level 2>,<level n>

We want to create a table of students and courses, but we have an array of course per each student

```
34 |   SELECT json_column:student.ID as student_id  
35 |   , json_column:courses as courses  
36 |   FROM students  
37 | ;  
38 |  
39 |  
40 |  
41 |  
42 |  
43 |  
44 |  
45 |
```

↳ Results ~ Chart

	STUDENT_ID	COURSES
1	1	[ "Math", "Science", "English", "History" ]
2	2	[ "English", "History" ]
3	3	[ "Math" ]
4	4	[ "English" ]

# TRANSFORMING JSON TO 3NF

We can select fields within the JSON using the following syntax:

<json column>:<level 1>,<level 2>,<level n>

We want to create a table of students and courses, but we have an array of course per each student

```
34 |   SELECT json_column:student.ID as student_id  
35 |   , json_column:courses as courses  
36 |   FROM students  
37 | ;  
38 |  
39 |  
40 |  
41 |  
42 |  
43 |  
44 |  
45 |
```

↳ Results ~ Chart

	STUDENT_ID	COURSES
1	1	[ "Math", "Science", "English", "History" ]
2	2	[ "English", "History" ]
3	3	[ "Math" ]
4	4	[ "English" ]

# TRANSFORMING JSON TO 3NF

In order to extract the array, we must use Lateral Flatten.

Syntax is the following:

**Select <reference column>, <Lateral Flatten Table>.VALUE**

**FROM <Source Table>,**

**Lateral Flatten(<column to flatten>)**

```
45  
46   SELECT json_column:student.ID as student_ID  
47   , X.VALUE as course  
48   FROM students,  
49   LATERAL FLATTEN(json_column:courses) X;  
50  
51
```

↳ Results ⚡ Chart

	STUDENT_ID	...	COURSE
6	2		"History"
7	3		"Math"
8	4		"English"
9	7		"Math"
10	7		"English"
11	7		"History"

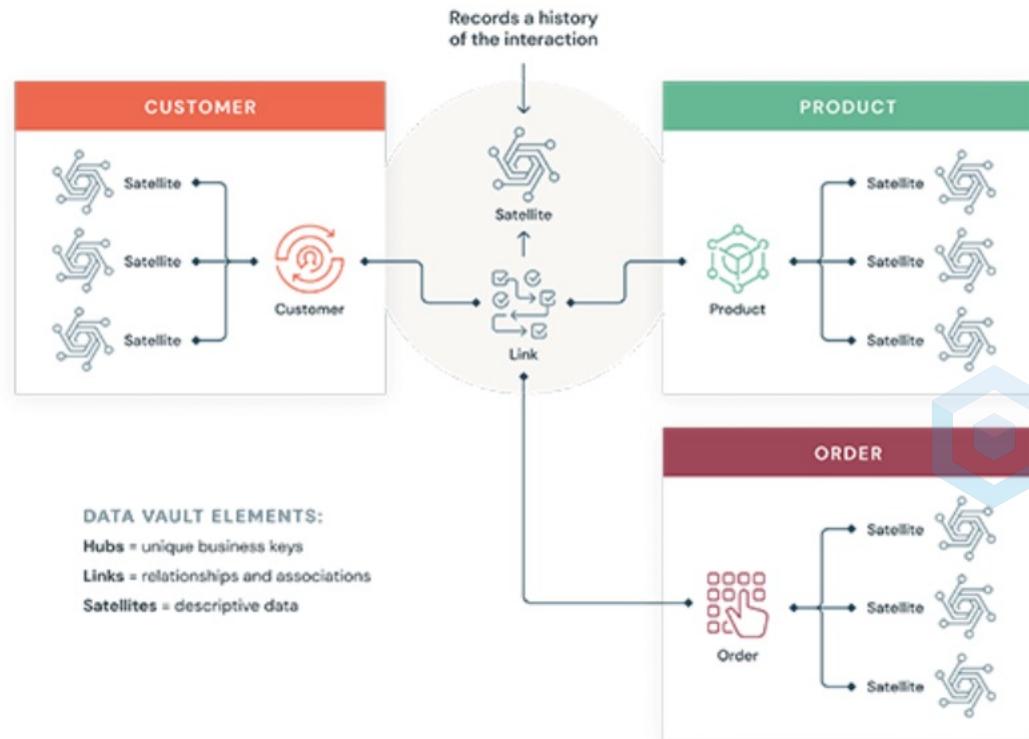


**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO DATA VAULT

Data Vault is an enterprise data model for robust analytics. The concept segregates key area into Hubs, with links showing the relationship between hubs called Links and objects within hubs called Satellites

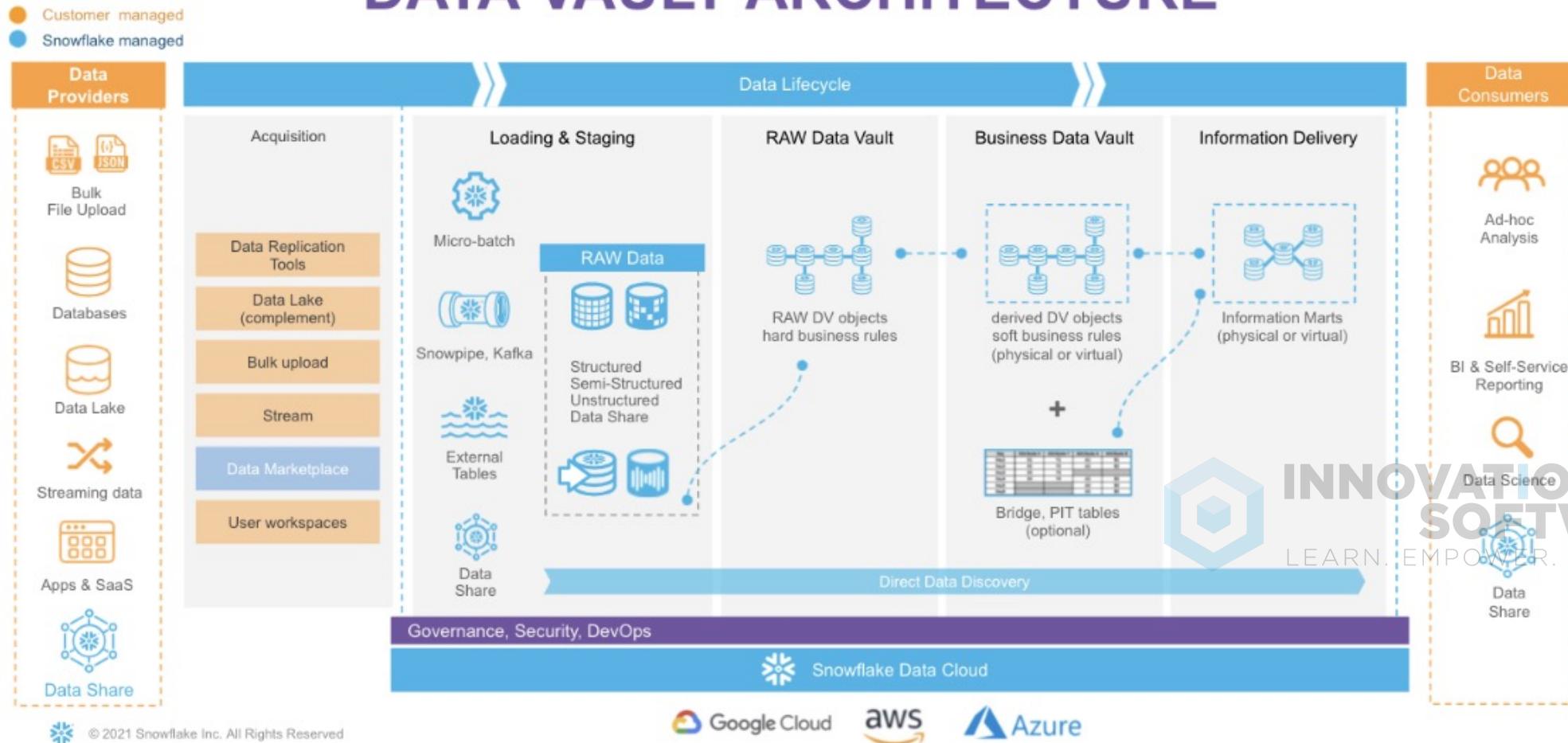
Data vault modeling



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO DATA VAULT

## MULTI-TIER DATA VAULT ARCHITECTURE



# TRANSFORMING JSON TO DATA VAULT

For this example, we want to convert the following JSON structure to a Data Vault.

We will separate the student, location and courses as their own hub.

Afterwards, we can create links and satellites with in Hubs.

```
student: {  
    "ID": 1,  
    "Name": "Jill",  
    "Age": 25  
},  
location: [  
    {  
        "name": "MI",  
        "active": "Y"  
    },  
    {  
        "name": "TX",  
        "active": "N"  
    }  
,  
courses: [  
    "Math",  
    "Science",  
    "English",  
    "History"  
]
```

# TRANSFORMING JSON TO DATA VAULT

According to the JSON, we are going to need three Hubs

```
26 | SELECT json_column:student  
27 | , json_column:courses  
28 | , json_column:location  
29 | FROM students;  
30 |
```

↳ Results    ↵ Chart

	JSON_COLUMN:STUDENT	JSON_COLUMN:COURSES	JSON_COLUMN:LOCATION
1	{ "Age": 25, "ID": 1, "Name": "Jill" }	[ "Math", "Science", "English", "History" ]	[ { "active": "Y", "name": "MI", "region": "N" }, { "active": "N", "name": "TX", "region": "S" } ]
2	{ "Age": 21, "ID": 2, "Name": "Mark" }	[ "English", "History" ]	[ { "active": "Y", "name": "TX", "region": "S" } ]
3	{ "Age": 18, "ID": 3, "Name": "Tim" }	[ "Math" ]	[ { "active": "Y", "name": "TX", "region": "S" }, { "active": "N", "name": "MI", "region": "N" } ]



INNOVATION  
SOFTWARE  
LEARN. EMPOWER. INNOVATE.

# TRANSFORMING JSON TO DATA VAULT

Focusing on just Student and Region, we can manipulate the JSON to have hubs for student and region, a satellite student table, with a link table

We can add additional satellite tables supporting each hub to provide more details

```
CREATE OR REPLACE TEMPORARY TABLE hub_student AS
SELECT json_column:student.ID as student_ID
FROM students
;

CREATE OR REPLACE TEMPORARY TABLE sat_student AS
SELECT json_column:student.ID as student_ID
, json_column:student.Name as name
, json_column:student.Age as age
FROM students
;

CREATE OR REPLACE TEMPORARY TABLE tmp_location_1 AS
SELECT json_column:student.ID as student_ID
, X.VALUE:name as state
, X.VALUE:region as region
, X.VALUE:active as active
FROM students,
LATERAL FLATTEN(json_column:location) X;

CREATE OR REPLACE TEMPORARY TABLE hub_state AS
SELECT state
, region
FROM (SELECT state, region FROM tmp_location_1 GROUP BY state, region) z
;

CREATE OR REPLACE TEMPORARY TABLE link_hub_location AS
SELECT student_ID
, state
, active
FROM tmp_location_1
;
```



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



**Schema on Read is when the schema is automatically inferred from the data sources:**

- A: True
- B: False

# POP QUIZ:

## DATA MODELING



**Schema on Read is when the schema is automatically inferred from the data sources:**

**A: True**

**B: False**

# POP QUIZ:

## DATA MODELING



**How do we unnest an array in Snowflake:**

- A: FLATTEN
- B: UNNEST
- C: UNARRAY
- D: LATERAL FLATTEN



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



**How do we unnest an array in Snowflake:**

- A: FLATTEN
- B: UNNEST
- C: UNARRAY
- D: LATERAL FLATTEN**



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



In a Data Vault, we have customers, orders and store locations. A table describing the orders per customer will be called a:

- A: Hub
- B: Link
- C: Satellite
- D: Mapping



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



In a Data Vault, we have customers, orders and store locations. A table describing the orders per customer will be called a:

A: Hub

B: Link

C: Satellite

D: Mapping



**INNOVATION**  
**SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



In a Data Vault, we have customers, orders and store locations. A table describing a customer addresses would be a:

- A: Hub
- B: Link
- C: Satellite
- D: Mapping



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# POP QUIZ:

## DATA MODELING



In a Data Vault, we have customers, orders and store locations. A table describing a customer addresses would be a:

- A: Hub
- B: Link
- C: Satellite
- D: Mapping



**INNOVATION  
SOFTWARE**  
LEARN. EMPOWER. INNOVATE.

# See you at Part 2!



INNOVATION  
SOFTWARE  
CREATE. EMPOWER. INNOVATE.