Snowflake SnowPro Core Certification Guide COF-C02

Hands-on exam preparation with practice questions

Balamurugan Kannaiyan

bpb

Snowflake SnowPro Core Certification Guide COF-C02

Hands-on exam preparation with practice questions

Balamurugan Kannaiyan

www.bpbonline.com

First Edition 2025

ISBN: 978-93-55518-880

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My beloved parents,
Mr. Kannaiyan *and* ***Mrs. Amsavalli****, and my loving family. I extend my sincere gratitude to my life partner* ***Jayashree****, adorable kids* ***Arnav*** *and* ***Aadith****, my dear sisters, and all my friends for their unwavering support throughout this incredible journey!*

About the Author

Balamurugan Kannaiyan is a highly accomplished Data Management and Cloud expert with over 18+ years of industry experience. With a distinguished track record, he leads the Data Engineering team, leveraging Snowflake's cutting-edge capabilities to build high-performance data applications. Bala brings extensive experience from prior roles within the Public Sector and Fortune 500 companies, where he excelled in designing scalable, cloud-distributed systems.

A graduate of the prestigious ANNA University, Bala holds a Bachelor of Engineering degree and numerous certifications and accreditations in cloud technologies and data management. As a Snowflake Squad Member, he actively participates in user group forums across the globe, sharing his expertise and insights with the community. He is passionate about travel, visiting new places, and gardening.

About the Reviewer

Pooja Kelgaonkar is a seasoned professional in the realm of data and a Snowflake data superhero for the years 2023 and 2024. With an impressive 18 years of industry experience exclusively dedicated to the data domain, Pooja has honed her expertise to become a trusted authority in Snowflake and data platform designs.

Currently serving as a Data Architect at Rackspace Technology, Pooja is stationed in the vibrant city of Toronto, Canada. Her role entails crafting intricate data architectures that drive innovation and efficiency for her clients. Pooja's profound understanding of Snowflake empowers her to architect robust solutions tailored to meet the diverse needs of businesses in the digital age.

Beyond her technical prowess, Pooja is recognized for her strategic mindset and ability to translate complex data challenges into actionable insights. Her dedication to staying at the forefront of emerging technologies ensures that she remains a trailblazer in the ever-evolving landscape of data management.

As a Snowflake data superhero, Pooja continues to inspire her peers and make significant contributions to the advancement of data-driven initiatives, cementing her legacy as a leader in the field.

Acknowledgement

First and foremost, I want to thank my parents and family from the bottom of my heart for their unwavering support and encouragement throughout the whole process of writing this book. I am so lucky to have you in my life.

To my mentors, I cannot thank you enough for all your guidance. I am grateful for everything you have taught me on this professional journey.

I would like to extend my sincere appreciation to the incredible team at BPB. Your support throughout the process of writing this book has been nothing short of amazing. Your dedication and commitment to excellence have been truly inspiring, and I am grateful for the opportunity to collaborate with such a talented group of individuals.

A special thanks to the technical reviewers Pooja Kelgaonkar and Sanjay Kattimani for the helpful suggestions that made this guide even better.

I am so grateful for my amazing colleagues in the Snowflake Data Superheroes and Squad. Their inspiration and support have been instrumental in my decision to share my knowledge and expertise with the broader community. I am honored to be part of this amazing group of experts who love what they do to help other data professionals.

Finally, I extend my sincere appreciation to the readers who have shown interest in this book. Your enthusiasm and support have been a driving force behind this project.

Preface

Snowflake is a cutting-edge, modern cloud data platform that is rapidly reshaping the landscape of data management and analytics. Organizations across industries increasingly adopt Snowflake to leverage its features to unlock the full potential of their data. This created a huge demand for skilled Snowflake professionals in the tech industry mastering Snowflake and SnowPro Certified. This book will be a comprehensive guide to help achieve the goal of everyone's SnowPro dream.

This book is for current and aspiring emerging Data professionals, Data/Solution Architects, Data Engineers, Database administrators, Data Analysts, Data scientists, and anyone who wants to explore and learn about the modern data cloud platform. This is an essential resource for any data professional seeking to master Snowflake and SnowPro Certified.

This book is divided into **15 chapters**. In this book, the reader will learn various topics, from Snowflake's fundamentals to advanced key concepts outlined in the SnowPro Certification. This comprehensive guide offers a detailed exploration of the Snowflake platform, empowering readers with the knowledge and skills required to design, implement, and manage robust cloud data solutions. Whether you are seeking to pass the SnowPro certification exam or enhance your expertise in a rewarding career, this book serves as an indispensable asset for your professional development.

Empower your career with this definitive guide to mastering Snowflake and become a SnowPro Certified to take advantage of the platform's fast-paced opportunities. In this book, you will learn the following:

Chapter 1: SnowPro Core Certification - In this chapter, we will provide an overview of SnowPro Core Certification, its prerequisites, and the target audience who will be the direct beneficiaries after completing this certification. We also learn about the list of overall domains, and particular domain topics to prepare and study for the SnowPro certification exam. In addition, we also learn how to register for exams, tips, and recommended resources to prepare and pass the exam.

Chapter 2: The Cloud Data Platform - In this chapter, we will learn about the Snowflake cloud data platform, Architecture, and technical details of the individual architecture components of the Service, Compute, and Storage layers. We also learn about the cloud providers, various editions, pricing details, Snowflake releases, and how Snowflake is different from legacy data warehousing solutions.

Chapter 3: Snowflake Cloud Data Platform Features - In this chapter, we will learn about the unique key features specific to Snowflake such as elastic storage, computing, and additional components within the data cloud ecosystems. We also learn about the impact and benefit of each Snowflake's key features.

Chapter 4: Snowflake Tools and User Interfaces - In this chapter, we will learn about how to get started with Snowflake, and step-by-step instructions to sign up Snowflake free trial account sign-up process. We also learned about the Snowflake user interfaces, Snowsight, Snowpark, and other technical details about various drivers and connectors available to connect Snowflake.

Chapter 5: Snowflake Catalogs and Objects - In this chapter, we will learn about the Snowflake Catalogs and objects such as databases, schemas, tables, views, and various data types available within Snowflake. In addition, we will learn more details about various other objects in Snowflake like Stored Procedures, User Defined Functions, Snowpipe, Streams, Tasks, Shares, Sequences, etc., to simplify and automate the data ingestion process.

Chapter 6: Snowflake Account Access - In this chapter, we will learn about the Snowflake Security principles, how to set up the Snowflake account access, and various network security policies to secure the account from unauthorized access. In addition, we will learn the details about the single sign-on (SSO), Multi-factor Authentication (MFA), and Federated authentication of Snowflake.

Chapter 7: Snowflake Security - In this chapter, we will learn about various Snowflake access control techniques like roles, and their hierarchy, privilege management, data governance tools, and so on. In addition, we will also learn the power of Snowflake's various data governance capabilities and how to secure the data.

Chapter 8: Snowflake Virtual Warehouse and Warehouse Management - This chapter aims to provide comprehensive information about Snowflake virtual warehouses, various characteristics, and configurations around the warehouse, scaling policy, data cache, how to manage and monitor the warehouses, and the credit usage and billing, etc. that helps to manage and work with the warehouse.

Chapter 9: Snowflake Performance Management - In this chapter, we will learn about various performance characteristics of Snowflake including the Query profile, Query history, characteristics of Query history, how to analyze and improve the query performance using various commands and techniques, etc.

Chapter 10: Snowflake Data Loading and Unloading - In this chapter, we will learn about essential aspects of Snowflake data movement, other technical components, and fundamental concepts like stages, file formats, file size considerations, and various methods and commands involved in this data movement process.

Chapter 11: Snowflake Data Transformations - In this chapter, readers will gain knowledge on various available data types in Snowflake, how to work with structured, semi-structured, and unstructured data, and other technical functionalities using UDF and stored procedures.

Chapter 12: Snowflake Data Protection - In this chapter, we will learn about how the data is secured and protected in Snowflake using various data protection features available in Snowflake such as Time Travel, Fail-safe, data encryption, cloning, replications, and further technical details.

Chapter 13: Snowflake Data Sharing - In this chapter, we will learn about the Snowflake Data Marketplace, Account Types, different data-sharing technologies, features, benefits, and how it works in real-time implementations.

Chapter 14: Snowflake Latest Additions - Snowflake keeps introducing new features to meet the industry demands. In this chapter, we will learn about the latest additions, such as Snowpark, Dynamic, Iceberg tables, and the Snowflake Cortex. We will explore how these innovations are continuously evolving to empower the Snowflake community and unlock the full potential of the platform for data management and analytics.

Chapter 15: Snowflake Knowledge Test - In this chapter, we have four sets of model questions to help the readers practice what they expect in the SnowPro Exam. Each model test consists of 25 questions to evaluate learner knowledge across different domains.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/ac420e

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Snowflake-SnowPro-Core-Certification-Guide-COF-C02.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
SnowPro Core Certification

Introduction

In this chapter, we will learn about the overview of SnowPro Core Certification, its prerequisites, and the target audience who will be the direct beneficiary after completing this certification. We also learn about the list of overall domains, and particular domain topics to prepare and study for the SnowPro certification exam. In addition, we will learn tips and recommended training to prepare and pass the exam.

Structure

The following are the topics to be covered:

- Certification overview
- Target audience
- Subject area breakdown
- SnowPro core domains and objectives
- Tips to prepare and pass the exam

Objectives

This chapter aims to provide overall information about the Snowflake SnowPro Core Certification and all the necessary technical details to prepare and pass the exam.

Certification overview

The SnowPro Core Certification became the most demanded certification across industries based on the latest trend. This certification demonstrates a person's knowledge and core knowledge to implement and migrate to Snowflake. This certification validates a candidate's understanding of Snowflake as a Data Cloud and how Snowflake can be used to drive business objectives. The SnowPro Core certification is a mandatory prerequisite for all advanced SnowPro Certifications and SnowPro Recertification.

This certification will test the ability to:

- Load and transform data in Snowflake
- Scale virtual warehouses for performance and concurrency
- Query concepts, DDL, and DML operations
- Work with semi-structured and unstructured data
- Utilize Snowflake's continuous data protection (cloning/time travel)
- Utilize data sharing
- Manage and monitor Snowflake accounts

Target audience

Anyone who wants to explore, learn about the modern data cloud and take advantage of the platform and fast-paced opportunities. We recommend that individuals with a minimum of six months of Snowflake experience and familiarity with basic SQL knowledge before attempting this exam:

- Solution architects
- Data engineers
- Data scientists
- Data analysts
- Snowflake account administrators
- Database administrators
- Application developers

Subject area breakdown

Snowflake continuously evolves as new features and functionalities are routinely developed to meet industry demands. The SnowPro Core exam (COF-C01) became available to the Snowflake community users in September 2019. A new version (COF-C02) of the exam was released in September 2022 with revisions made to the exam contents for modernization purposes.

If you are already SnowPro Core certified, please note that you will not be required to take the new version of the exam to maintain your badge or certification status. However, you will follow the new version of the exam for recertification two years after you originally passed the exam. The exam formats for both first and recertification remain the same. Refer to the following details for additional reference:

	SnowPro Core	Recertification - SnowPro Core
Prerequisites	None	SnowPro Core Certification
Exam Fees	USD 175	USD 88
Number of Questions	100	60
Question Types	Multiple Select/Choice True or False	Multiple Select/Choice True or False
Time Limit	115 Minutes	85 Minutes
Passing Score **(Scaled Scoring from 0-1000)**	750	750
Exam Delivery Options	Online Proctoring & Onsite Testing Centers	Online Proctoring & Onsite Testing Centers
Languages	English Japanese	English Japanese (Release date to be announced)

Figure 1.1: *SnowPro Core - Certification Vs Recertification*

Overall, the initial and newer versions (revised in September 2022) of the SnowPro exam domains remain the same, and revisions are made to exam contents for modernization purposes. A few of the task objectives have been eliminated from the updated content outline as they covered topics that were either already sufficiently covered in other areas of the blueprint or are no longer relevant.

Please refer to the following information for what changes to expect in the revised version of the SnowPro Exam. The alignment and weightings for the main content domains from both exam blueprints are listed for comparison purposes below:

COF-CO1 Domain (SnowPro Prior Exam)	Estimated % Range
Snowflake Overview & Architecture	25 – 30 %
Account and Security	10 – 15 %
Performance Management	05 – 10 %
Data Movement	11 – 20 %
Virtual Warehouses	15 – 20 %
Storage and Protection	10 – 15 %

COF-CO2 Domains (SnowPro Revised Exam)	Estimated % Range
Snowflake Cloud Data Platform Features and Architecture	20 – 25 %
Account Access and Security	20 – 25 %
Performance Concepts	10 – 15 %
Data Loading and Unloading	5 – 10 %
Data Transformations	20 – 25 %
Data Protection and Data Sharing	5 – 10 %

Figure 1.2: *SnowPro Core – prior vs latest version: list of domains and estimated questions range*

SnowProcore domains and objectives

This exam outline includes test domains, weightings, and objectives. Please refer to the contents below for the topics required to study for the individual domains:

- **Domain: Snowflake cloud data platform features and architecture**
  - Outline key features of the Snowflake Cloud Data Platform
    - Snowflake's key architecture layers
    - Elastic Compute
    - Elastic Storage
    - Cloud partner categories
    - The Data Cloud / Data Exchange / Partner Network
  - Outline key Snowflake tools and user interfaces
    - Snowflake user interfaces (Web UI)
    - Classic vs. Snowsight
    - Snowflake drivers
    - Snowflake connectors
    - Snowpark
    - SQL scripting
  - Outline Snowflake's catalogs and objects
    - Snowflake databases
    - Snowflake schemas
    - Data types

        - Tables and table types
        - Views and view types
        - **User defined functions (UDFs)**
        - **User defined table functions (UDTFs)**
        - **Stored procedures (SPs)**
        - Pipes
        - Streams and tasks
        - Shares
        - Sequences
    - Outline Snowflake Storage Structure
        - Column metadata clustering
        - Micro partition
        - Storage monitoring
        - Search optimization
- **Domain: Account access & security**
    - Outline Security Concepts
        - **Single Sign-On (SSO)**
        - **Multi-Factor Authentication (MFA)**
        - Network policies and security
        - Federated authentication
    - Define the roles and entities used in Snowflake
        - Explain roles hierarchy and privilege offerings
        - Define how privileges are granted and revoked
    - Snowflake - Data governance capabilities
        - Snowflake account
        - Organization and hierarchy
        - Snowflake database
        - Information schemas
        - Snowflake Secure views
        - Access history and read support
- **Domain: Snowflake performance concepts**
    - Describe the use of the query profile

        - Use of the data cache
        - Query plans
        - Data spilling
        - Micro-partition and pruning
        - Query history
    - Describe Virtual Warehouses (WH) and configurations
        - Single vs. multi cluster
        - Warehouse size
        - Warehouse setting
        - Warehouse access
    - Virtual warehouse performance tools
        - Monitor warehouse loads
        - Scale up vs. scaling out
        - Query performance
        - Resource monitoring
    - Optimize query performance
        - Use of specific **`SELECT`** commands
        - Materialized views
- **Domain: Data loading and unloading**
    - Define best practices and concepts that should be considered when loading data
        - Stage and stage types
        - File formats
        - File size
        - Folder structures
        - Ad hoc/bulk loading using the Snowflake UI
    - Outline different commands used to load data and when they should be used
        - Copy into
        - Create pipe
        - Get
        - Put
        - Insert/Insert overwrite

        - Snowflake stream
        - Task
        - Validate
    - Define concepts and best practices that should be considered when unloading data
        - Snowflake file formats
        - NULL values and empty strings
        - Unloading to a file
        - Unloading relational tables
    - Outline the different commands used to unload data and when they should be used
        - Copy into
        - List
        - File format creation
        - Create file format and clone
        - Describe file format
        - Alter file format
        - Show file format
        - Drop file format
    - Describe how to load and work with semi-structured data.
        - Supporting file formats, data types, and file size
        - VARIANT column/data type
        - Flattening nested structure of data
- **Domain: Data transformations**
    - Describe how to work with standard data
        - Supported function types
        - Estimation functions
        - Sampling
        - **User defined functions (UDFs)**
        - Stored procedures
    - Explain how to work with semi-structured data
        - VARIANT column
        - Supported file formats, data types, and sizes

        - Flattening the nested structure
    - Explain how to work with unstructured data
        - Describe and use directory tables
        - SQL file functions
        - Purpose of UDFs for data analysis
- **Data protection and data sharing**
    - Outline Snowflake continuous data protection
        - Snowflake time travel
        - Snowflake failsafe
        - Cloning
        - Data encryption
        - Replication
    - Outline Snowflake data sharing capabilities
        - Snowflake account types
        - Private data exchange
        - Data marketplace and data exchange
        - Shares
        - Access control options

Tips to prepare and pass the SnowProcore exam

Individuals should have at least 6 months of knowledge using Snowflake before attempting this exam. We recommend the individuals a combination of hands-on experience, and instructor-led/on-demand training plus using this guide as self-study material will help to pass the exam easily.

Recommended training materials and hands-on reference to this self-study guide:

- This exam guide and practice test
- Snowflake university on-demand training
- Snowflake 30-day free trial account for hands-on lab
- Getting started with Snowflake - Zero to Snowflake
- Snowflake documentation
- Stay connected with the local Snowflake community
- Make sure you stay up to date on the latest product releases

Exam registration instructions

Snowflake's SnowPro Certification exams are delivered through Pearson Vue and can be taken at any of the 1000+ testing centers located globally or remotely in your home with a virtual proctor.

As a participant in the SnowPro certification program, candidates are required to review and accept Snowflake's certification terms and conditions during the registration process. Failure to agree to these terms will not allow you to register for any Snowflake exam.

Creating an account with Snowflake Certification Portal or Pearson Vue:

- Determine if you want to take your exam in person at a testing Center or online. (See onsite proctored or online proctored requirements for additional information.)
- If you are a first-time user **Create New Account** to register or existing users can log in using the credentials to access the following Snowflake Certification Portal | **https://cp.certmetrics.com/snowflake/en/login**

  The following figure illustrates the Snowflake Certification Portal Login Screen:

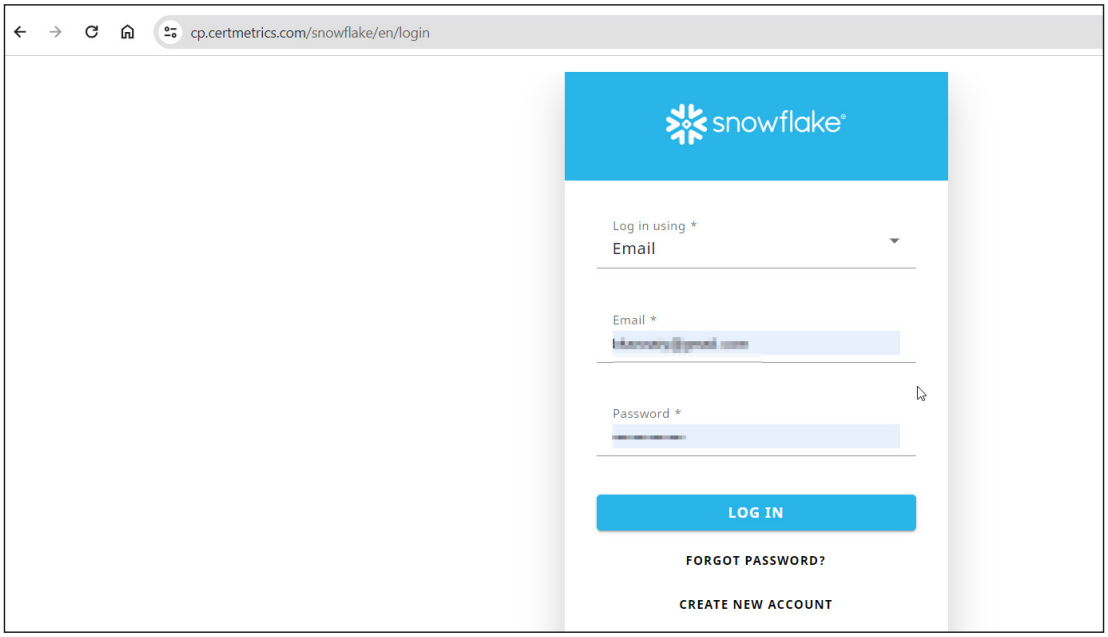

Figure 1.3: *SnowPro Certification Registration/Login Page*

- Once logged into the certification portal, navigate using the left page navigation section and browse to **SCHEDULE AND MANAGE EXAMS** and then **Manage My Exams** sub-section to schedule the SnowPro Core Certification exam. This link takes the users to the Pearsonvue web page.

The following figure illustrates the Snowflake Certification Portal | **Schedule and Manage** section:

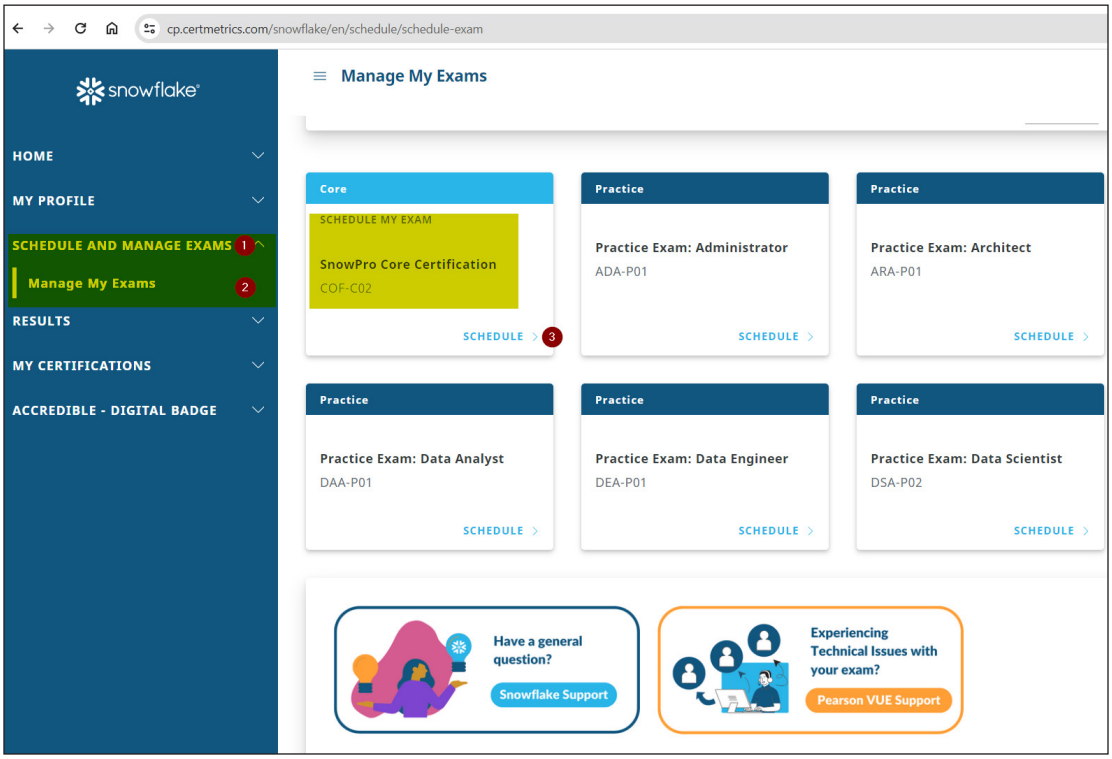

Figure 1.4: *SnowPro Certification Registration/Scheduling – Snowflake Portal Page*

- Select the delivery option, date, and time you want to take the exam. You may choose from **In Person at a test center** or **Online with OnVUE** based on a suitable option and convenience. Be sure to review the reschedule and cancellation policy and agree to the Snowflake Certification Agreement.

  The following figure illustrates the Pearson Portal page redirected from the prior step:

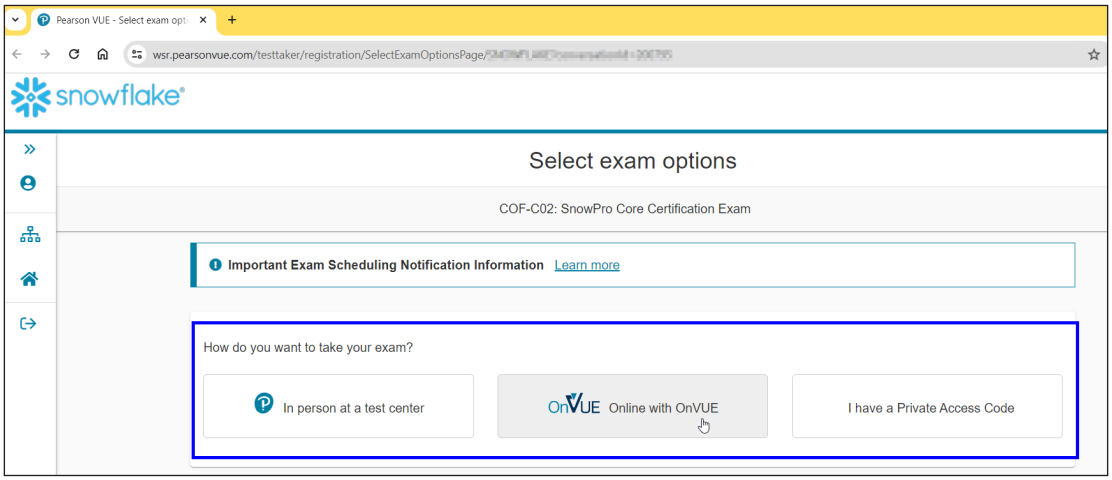

Figure 1.5: *SnowPro Certification Registration/Scheduling – Pearson Vue Page*

- Review and respond to Pearson Vue's Privacy Policy Acceptance terms
- Time zone selection will be based on the address entered on your profile.
- Follow the payment instructions and click **Submit**.

  If you annot decide between going into a testing Center or taking the exam at home with an online proctor? Check out PearsonVue's helpful resources for test takers (**https://home.pearsonvue.com/Test-takers/Resources.aspx**).
- Do not forget to review SnowPro Program Policies before your exam.

Refer to the following web link for additional instructions for the Snowflake Certification portal or Pearson Vue's Registration process:

https://learn.snowflake.com/courses/course-v1:snowflake+CERT-EXAM-REG+A/about

Conclusion

In this chapter, we learned about the required technical contents for SnowPro Core Certification, list of overall domains, and individual domain topics to prepare and study for the SnowPro certification exam. We also learned the tips and recommended training to prepare and pass the exam.

Let us continue to read the next chapters to learn about the technical information for individual domains.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 2
The Cloud Data Platform

Introduction

In this chapter, we will learn about the Snowflake cloud data platform, Architecture, and technical details of the individual architecture components of the Service, Compute, and Storage layers. We also learn about the cloud providers, various editions, pricing details, and how Snowflake is different from legacy data warehousing solutions.

Structure

The following are the topics to be covered:

- Platform overview
- Snowflake data cloud key capabilities
- Cloud offering
- Non-cloud and cloud offering: High-level overview
- Architecture components
- Snowflake three layers
- Cloud providers
- Snowflake editions

- Snowflake releases
- Snowflake pricing

Objectives

This chapter aims to provide technical details about the Snowflake Cloud Data platform, Architecture components, and key components within the architecture, various editions, and cloud providers that Snowflake supports to meet the user/industry needs.

Platform overview

The Snowflake Cloud Data platform was founded by *Benoit Dageville*, and *Thierry Cruanes* in July 2012, later it was publicly launched after two years in October 2014. *Benoit* was given an opportunity by an early investor to rearchitect the data warehouse in the cloud entirely from scratch. They repurposed a brand-new architecture for the cloud, which is designed to support fault isolation, performance isolation, and elasticity of the cloud. With these requirements in hand, *Benoit* and *Thierry* built the system *Snowflake* that would adopt the resources based on the demand and scale to infinity, and the customers only pay for what they use.

The Snowflake Data Cloud offers a cloud-based analytics and data storage solution also known as **Data Warehouse as a service**. This is an advanced platform delivered as a self-managed service. The Snowflake enables the organization to build modern data warehousing solutions in the cloud (data storage, processing, and analytics) in a faster, easier, and more flexible style compared to traditional data warehousing solutions built in a few hours instead of waiting for months to get ready.

The Cloud Data Platform: How it is a self-managed service?

- No hardware to select, install, manage, or configure (Virtual/Physical)
- No software to select, install, manage, or configure (Virtual/Physical)
- No infrastructure or ongoing maintenance, upgrades, management, and fine-tuning
- It is not a package software that the users require to be installed

Overall, Snowflake manages all aspects of hardware, software, upgrades, and configurations behind the scenes without any user interventions. This allows the customers to store and analyze data in the cloud without managing the hardware, software, or infrastructure additional overhead to manage the platform.

Snowflake offers all three major cloud providers to choose from various editions and features. In the very beginning, Snowflake's release was offered in **Amazon Web Services** (**AWS**), but later Snowflake extended its offering in Azure and GCP Cloud to meet the industry demand. This is another versatile feature to pick Snowflake. The customers have the flexibility to choose one or more cloud providers based on their need to set up the Snowflake Account.

Snowflake data cloud key capabilities

The Snowflake is a modern data warehousing solution built on the cloud from scratch. The Snowflake offered as a **Software-as-a-Service** (**SaaS**) aims to be a one-stop solution to support various modern data warehousing use cases like data engineering, data lake, data science and ML, data applications, data exchange, and much more. Snowflake's unique elasticity supports any scale of data, processing, and workloads.

The following figure illustrates various use cases of Snowflake and how the data is shared across different requirements. It is like storing the data once and sharing as many as you want.

Figure 2.1: *Snowflake Platform and its offering (Source: Snowflake documentation)*

- **Data Warehouse as a Service**: Snowflake allows you to focus on collecting, unifying, and using your data. The rest will be taken care of by Snowflake. Snowflake's Data warehouse as a service eliminates the need to deploy and manage hardware or software. This also comes with built-in enterprise-level availability, end-to-end encryption, data protection, and no manual knobs to tune.
- **Data sharing for all**: The traditional data-sharing methods present significant challenges in terms of complexity and cost, even when transferring a limited portion of data from a data warehouse. Today's volume variety and velocity of data only intensify sharing issues. To solve these critical challenges, Snowflake innovated its build for the cloud data warehouse to the idea of Snowflake Data sharing. Store once and share with many.
- **Multi-dimensional elasticity**: Snowflake's unique Architecture enables it to support any scale of storage, computing, and users. Scale storage to match any

volume of data and adjust your compute up or down on the fly even automatically without disruption to the existing processes. This provides flexibility to create additional compute clusters to support as many workloads or users as you need without moving the copying of the data. The customers can deliver the perfect amount of resources at the exact time they are needed.

- **All business data in one place**: Snowflake supports Structured and semi-structured data in a single system. Snowflake supports loading semi-structured data to a new column data type VARIANT. Users can directly load the semi-structured data such as JSON, AVRO, Parquet, or ORC files without any additional transformation and query both structured and semi-structured data even in a single query.

Cloud offering

You might already heard about various Cloud offerings. However, refer to the below details about various Non-Cloud and Cloud offerings and how they differ from each other.

Each cloud model offers unique specific features and functionality, so it's critical to understand what model fits your organization's use cases. Also, no matter what option you choose, Cloud migration is the latest technology trend to upgrade your platform to meet the demands of any business.

The following are the types of non-cloud and cloud offering:

- **On-prem (non-cloud):** This is the typical type of service most companies used to perform in the past where everything is managed by the company on its own hardware/software/patching and so on. This is outside of the cloud offering, just to provide insight to the users.

  Common examples: on-prem database, applications, and so on.

- **Infrastructure as a Service (IaaS): Infrastructure as a Service** (**IaaS**) provides virtualized computing infrastructure, including servers, networks, operating systems, and storage, through a cloud delivery model. It delivers a foundational IT environment, provisioned by the vendor or a third party, ready for the customer to deploy and configure their own applications and services.

  Common examples: AWS, Microsoft Azure, GCP, Rackspace, and so on.

- **Platform as a Service (PaaS):** PaaS offers necessary hardware and software resources by the vendor or their provider in the cloud. These offerings enable the customers to develop, run, and manage business applications without managing the infrastructure required for the development process.

  Common examples: SAP Cloud, Microsoft Azure, Aws Lambda, and so on.

- **Software as a service (SaaS):** SaaS is the most utilized option in the Cloud Market. The majority of the service is managed by the cloud providers and delivered as

a packaged solution. With this model; the customers can focus on delivering the business priorities instead spend time managing the hardware or infrastructure.

Common examples: Snowflake, DropBox, Salesforce, and so on.

Non-cloud and cloud offering: High-level overview

The following figure illustrates the high-level overview of non-cloud and various cloud offerings and how its managed by the provider or customer:

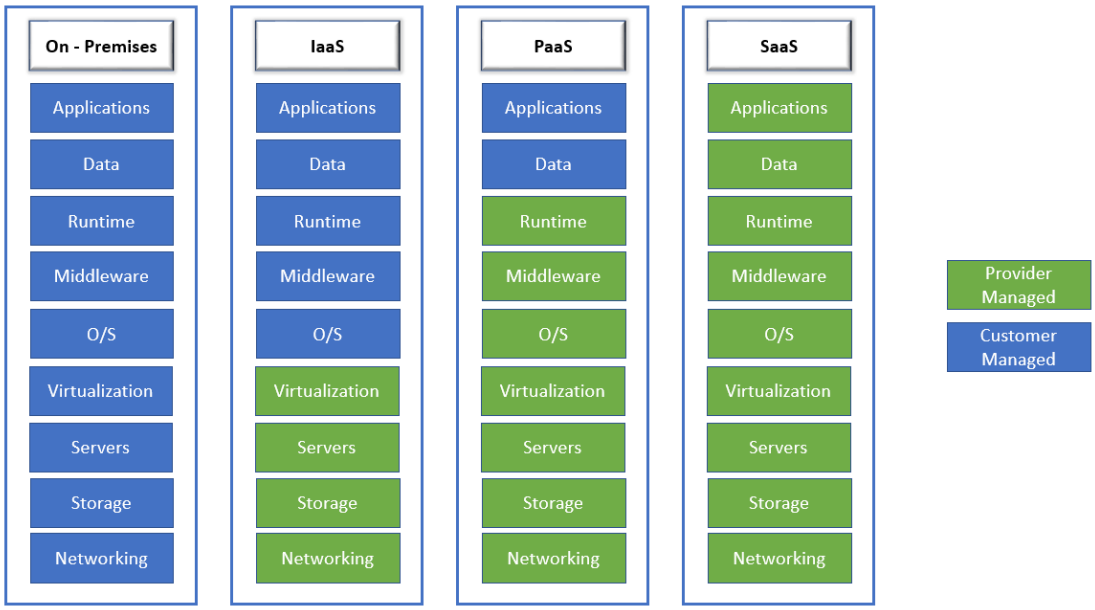

Figure 2.2: *Non-cloud and cloud offering models*

Architecture components

The Snowflake modern cloud data platform was designed from scratch. This is not built on any existing technology or database or Big Data platform like Hadoop. Snowflake is built on a completely new SQL query engine that is natively architecture and designed for the cloud.

The unique feature of the Snowflake architecture to the separation of storage and computing compared to other traditional data warehousing solutions. Technically it is like the combination of shared disk and shared nothing database architecture. This new feature allows the organization/user to scale compute on the fly and store an infinite amount of data instantly.

Refer to the following figure for the key Snowflake layers and how it is organized within the Snowflake architecture:

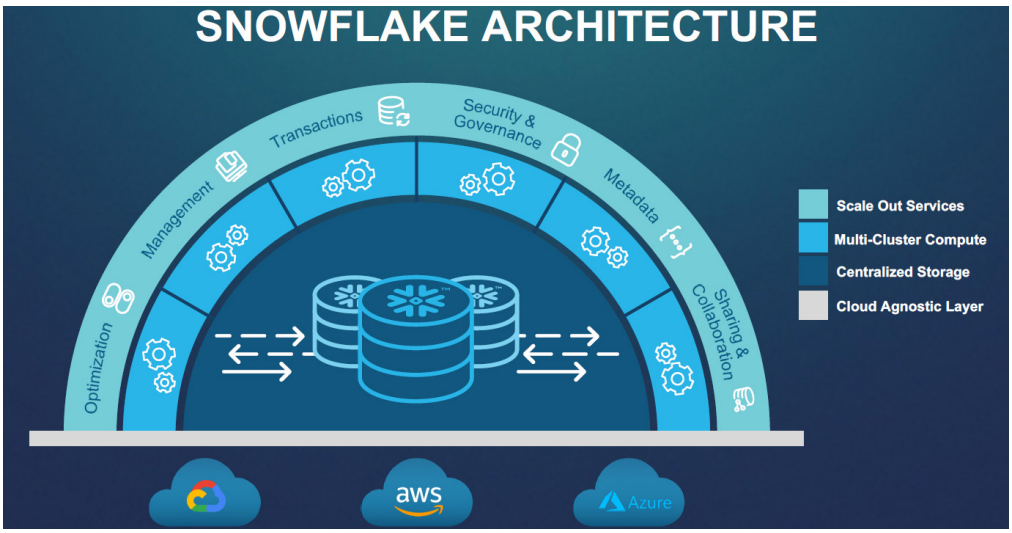

Figure 2.3: *Snowflake's architecture and its layers (Credit: Snowflake documentation)*

The Snowflake architecture build-up of the following key components or layers, as shown in *Figure 2.3*:

- Cloud service layer
- Compute or Query processing layer
- Database storage layer
- Cloud agnostic layer

Let us understand the role of each layer and what functionality/role each of these layers plays in Snowflake architecture.

Snowflake three layers

From Snowflake's architecture, the key layers are Service, Compute, and Data Storage. Each layer plays a very important role in architecture:

- **Cloud Service or Service Layer**: The Cloud service layer is also known as the Brain of Snowflake. This layer handles overall Snowflake account management and collection of services interacting with other layers to support Snowflake usage. The Service layer manages and controls activities like account access, security, user authentication, governance, metadata management, infrastructure management, query optimization, etc.
- **Compute layer**: The compute layer is also known as the Query Processing Layer. The Compute layer manages and controls all the query execution using the *Virtual Warehouse*. The virtual warehouse relates to the compute, and this can scale up or down instantly at any time. We will learn more about Virtual Warehouse in the upcoming chapters.

- **Database storage layer**: This layer is also known as Storage Layer. When data gets loaded into Snowflake, The Storage layer manages, organizes, and stores the data (structured, semi-structured, unstructured, and so on) in its optimized, compressed format. The data will be reorganized and stored in columnar format for faster retrieval. All aspects of the data storage are handled by Snowflake. The data objects stored inside Snowflake are not directly accessible or visible to the customers, its only accessible thru SQL query operations using Snowflake.
- **Cloud agnostic layer**: The Cloud Agnostic layer is about which cloud provider the Snowflake account is hosted on. Snowflake supports the following cloud providers based on customer choice:
    - **Amazon Web Services** (**AWS**)
    - Microsoft Azure
    - **Google Cloud Platform** (**GCP**)

Cloud providers

Snowflake provides complete flexibility to choose the cloud providers based on enterprise technology stack and need. The Snowflake account can host in any of the following cloud providers. Each platform provides one or more regions where the Snowflake account is provisioned based on the customer's choice.

- **Amazon Web Services** (**AWS**) (2014 Onwards)
- **Microsoft Azure** (2018 onwards)
- **Google Cloud Platform** (**GCP**) (2019 Onwards)

If your organization's services are hosted in one of the platforms, it is best to practice choosing the same cloud platform to host your Snowflake account. However, the customer has the option to choose a different cloud platform to host the Snowflake account.

Snowflake was initially released in AWS and Azure platforms; later in 2020, GCP was also added to the flavor to provide additional flexibility to the customers to choose the platform of their choice.

Snowflake editions

Snowflake offers various editions to fit the organization's and users' specific needs, this enables customers can choose the editions that satisfy their requirements. As your organization grows and changes in needs, customer has the flexibility to change the editions as an easy option.

Snowflake provides the following editions at a high level:

- **Standard edition**: This is Snowflake's introductory level offering, which provides unlimited access to Snowflake's standard features. Best for small to medium-level organization use cases.

- **Enterprise edition**: This is the second-level edition of Snowflake. Enterprise editions provide all features and services of Standard Edition Plus additional features specifically designed for large-scale enterprise organizations
- **Business critical edition**: The next level to the Enterprise edition is Business Critical. This version is also formally known as **Enterprise for Sensitive Data** (**ESD**). Business Critical edition offers higher data protection than Standard/Enterprise to support organization that handles very sensitive data specifically PII and PHI data that require various regulations like HIPPA, HITRUST, and so on.

  This edition includes all features and services of Enterprise Edition plus enhanced data protection and security. Also adds database failover/failback to support business-critical application continuity.
- **Virtual Private Snowflake (VPS)**: Virtual Private Snowflake edition is the topmost offering that provides the highest level of security for any enterprise or organization that handles highly sensitive data such as banking, financials, or similar large enterprise organizations that handle extremely sensitive data.

The Virtual Private Snowflake edition includes all features and services of Business-Critical Edition plus setup in a completely different environment isolated from other Snowflake accounts. The VPS Snowflake account doesn't share any environment or resources outside VPS.

Snowflake editions and features for quick reference

The following are the snowflake editions for quick reference:

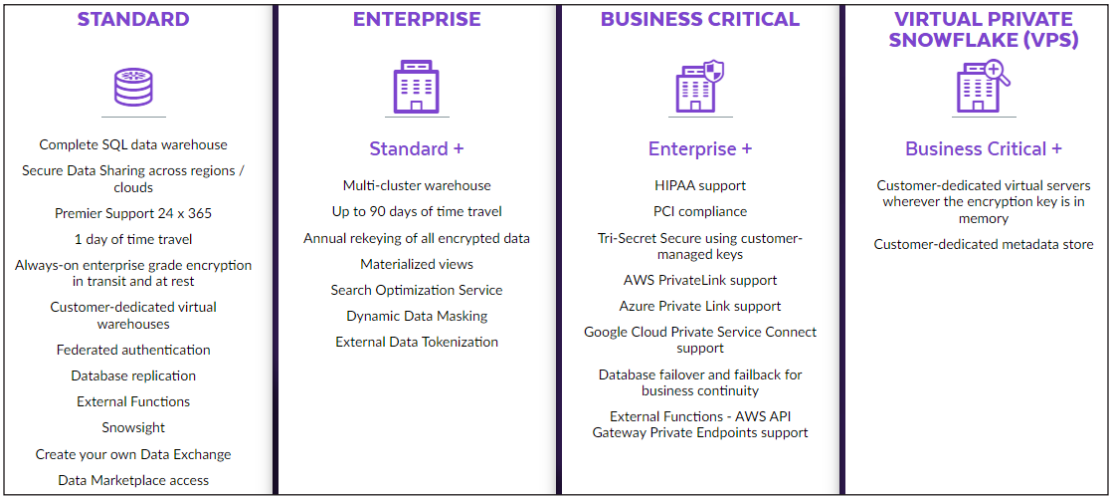

Figure 2.4: *Snowflake various editions and features (Source: Snowflake documentation)*

Snowflake editions/features matrix

Refer to the following table for the list of major features/services related to release management, security, data governance, and protection for the individual edition of Snowflake:

Release Management				
Feature/Service	Standard	Enterprise	Business Critical	VPS
24-hour early access to weekly new releases, which can be used for additional testing/validation before each release is deployed to your production accounts		✓	✓	✓
Security, Governance, & Data Protection				
Feature/Service	Standard	Enterprise	Business Critical	VPS
SOC 2 Type II certification	✓	✓	✓	✓
Federated authentication and SSO for centralizing and streamlining user authentication	✓	✓	✓	✓
OAuth for authorizing account access without sharing or storing user login credentials	✓	✓	✓	✓
Network policies for limiting/controlling site access by user IP address	✓	✓	✓	✓
Automatic encryption of all data	✓	✓	✓	✓
Support for multi-factor authentication	✓	✓	✓	✓
Object-level access control	✓	✓	✓	✓
Standard Time Travel (up to 1 day) for accessing/restoring modified and deleted data	✓	✓	✓	✓
Disaster recovery of modified/deleted data (for 7 days beyond Time Travel) through Fail-safe	✓	✓	✓	✓
Extended Time Travel (up to 90 days)		✓	✓	✓
Periodic rekeying of encrypted data for increased protection		✓	✓	✓
Column-level Security to apply masking policies to columns in tables or views		✓	✓	✓
Row Access Policies to apply row access policies to determine which rows are visible in a query result		✓	✓	✓
Object Tagging to apply tags to Snowflake objects to facilitate tracking sensitive data and resource usage		✓	✓	✓
Support for classifying potentially sensitive data using classification		✓	✓	✓
Audit the user access history through the Account Usage ACCESS_HISTORY view		✓	✓	✓
Customer-managed encryption keys through Tri-Secret Secure			✓	✓
Support for Private Connectivity to the Snowflake Service using AWS PrivateLink, Azure Private Link, or Google Cloud Private Service Connect			✓	✓
Support for Private Connectivity to Snowflake Internal Stages using AWS PrivateLink and Azure Private Link			✓	✓
Support for PHI data (in accordance with HIPAA and HITRUST CSF regulations)			✓	✓
Support for PCI DSS			✓	✓
Support for FedRAMP Moderate data (in the US government regions)			✓	✓
Support for IRAP - Protected (P) data (in specified Asia Pacific regions)			✓	✓
Dedicated metadata store and pool of compute resources (used in virtual warehouses)				✓

Figure 2.5: *Snowflake edition/features matrix – release management and security*

Refer to the following table for the list of major features/services related to Compute, SQL support, and data import/export for the individual edition of Snowflake:

Compute Resource Management				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Virtual warehouses, separate compute clusters for isolating query and data loading workloads	✓	✓	✓	✓
Resource monitors for monitoring virtual warehouse credit usage	✓	✓	✓	✓
Multi-cluster virtual warehouses for scaling compute resources to meet concurrency needs		✓	✓	✓
SQL Support				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Standard SQL, including most DDL and DML defined in SQL:1999	✓	✓	✓	✓
Advanced DML such as multi-table INSERT, MERGE, and multi-merge	✓	✓	✓	✓
Broad support for standard data types	✓	✓	✓	✓
Native support for semi-structured data (JSON, Avro, ORC, Parquet, and XML)	✓	✓	✓	✓
Native support for geospatial data	✓	✓	✓	✓
Native support for unstructured data	✓	✓	✓	✓
Collation rules for string/text data in table columns	✓	✓	✓	✓
Integrity constraints (not enforced) on table columns for informational and modeling purposes	✓	✓	✓	✓
Multi-statement transactions	✓	✓	✓	✓
User-defined functions (UDFs) with support for Java, JavaScript, Python, and SQL	✓	✓	✓	✓
External functions for extending Snowflake to other development platforms	✓	✓	✓	✓
Amazon API Gateway private endpoints for external functions			✓	✓
Stored procedures with support for Java, JavaScript, Python, Scala, and SQL (Snowflake Scripting)	✓	✓	✓	✓
External tables for referencing data in a cloud storage data lake	✓	✓	✓	✓
Support for clustering data in very large tables to improve query performance, with automatic maintenance of clustering	✓	✓	✓	✓
Query accleration for parallel processing portions of eligible queries		✓	✓	✓
Search optimization for point lookup queries, with automatic maintenance		✓	✓	✓
Materialized views, with automatic maintenance of results		✓	✓	✓
Data Import & Export				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Bulk loading from delimited flat files (CSV, TSV, etc) and semi-structured data files (JSON, Avro, ORC, Parquet, and XML)	✓	✓	✓	✓
Bulk unloading to delimited flat files and JSON files	✓	✓	✓	✓
Snowpipe for continuous micro-batch loading	✓	✓	✓	✓
Snowflake Connector for Kafka for loading data from Apache Kafka topics	✓	✓	✓	✓

Figure 2.6: *Snowflake edition/features matrix – Compute, SQL support, and data import/export*

Refer to the following table for the list of major features/services related to tools, interfaces, data, and customer support for the individual edition of Snowflake:

Interfaces & Tools				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Snowsight, the next-generation SQL worksheet for advanced query development, data analysis, and visualization	✓	✓	✓	✓
SnowSQL, a command line client for building/testing queries, loading/unloading bulk data, and automating DDL operations	✓	✓	✓	✓
SnowCD, a command line diagnostic tool for identifying and fixing client connectivity issues	✓	✓	✓	✓
Programmatic interfaces for Python, Spark, Nodejs, and Go	✓	✓	✓	✓
Native support for JDBC and ODBC	✓	✓	✓	✓
Extensive ecosystem for connecting to ETL, BI, and other third-party vendors and technologies	✓	✓	✓	✓
Snowflake Partner Connect for initiating free software/service trials with a growing network of partners in the Snowflake ecosystem	✓	✓	✓	✓
Snowpark, a library that provides an intuitive API for querying and processing data in a data pipeline	✓	✓	✓	✓
Data Pipelines				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Streams for tracking table changes	✓	✓	✓	✓
Tasks for scheduling the execution of SQL statements, often in conjunction with table streams	✓	✓	✓	✓
Data Replication & Failover				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Database and share replication between Snowflake accounts (within an organization) to keep database and share objects and stored data synchronized	✓	✓	✓	✓
Failover and failback between Snowflake accounts for business continuity and disaster recovery			✓	✓
Redirecting Client Connections between Snowflake accounts for business continuity and disaster recovery			✓	✓
Data Sharing				
Feature/Service	Standard	Enterprise	Business Critical	VPS
As a data provider, securely share data with other accounts	✓	✓	✓	
As a data consumer, query data shared with your account by data providers	✓	✓	✓	
Secure data sharing across regions and cloud platforms (through data replication)	✓	✓	✓	
Snowflake Marketplace and Listings, where providers and consumers meet to share data securely	✓	✓	✓	
Data Exchange, a private hub of administrators, providers, and consumers that you invite to securely collaborate around data	✓	✓	✓	
**** By Default, Snowflake VPS doesn't allow Data sharing outside of the VPS Account.**				
Customer Support				
Feature/Service	Standard	Enterprise	Business Critical	VPS
Snowflake Community, Snowflake's online Knowledge Base and support portal (for logging and tracking Snowflake Support tickets)	✓	✓	✓	✓
Premier support, which includes 24/7 coverage and 1-hour response window for Severity 1 issues	✓	✓	✓	✓

Figure 2.7: *Snowflake edition/features matrix – tools/interfaces, data features and customer support*

Snowflake releases

Snowflake wants to make sure the users always have the latest features and bug fixes to keep things running smoothly. To do this, Snowflake releases new features and updates frequently, typically every week. These updates happen in the background, so users won't experience any downtime and will always have the latest version. Some accounts can even get early access to new features for testing purposes. Following are the two main types of releases:

- **Regular releases (weekly)**: These releases introduce new features, enhancements, and bug fixes. They typically don't contain any breaking changes to how Snowflake works.
- **Behavior change releases (monthly)**: One release each month (except November and December) is chosen to potentially include changes that might affect how your existing Snowflake code works. These changes are usually minor, but it's a good idea to be aware of them.

Here is a simplified breakdown of what you can expect in each type of release:

- **Regular releases**: Look out for new features that can expand what you can do with Snowflake, along with improvements and bug fixes that make Snowflake run smoother.
- **Behavior change releases**: Be on the lookout for announcements about any changes that might require adjustments to your existing Snowflake code. These are usually minor but important to be aware of.

For specific details about what's new in each release, you can check the Snowflake documentation **https://docs.snowflake.com/en/release-notes/new-features**. This link has information on recent releases, including what type of release it was and what the key changes are, and so on.

Snowflake pricing

The Snowflake pricing is based on the volume of data that you store in Snowflake and the compute time. The data cloud offers either on-demand or pre-purchased capacity options with usage-based per-second pricing. The customer pays only for what they use for computing and storage.

Another factor that influences the cost model will be where the Snowflake account is hosted (Cloud provider/region) and what type of storage plan (on-demand or capacity storage). Let us do a deeper dive in the next section on how the actual Snowflake billing is calculated.

Storage pricing plans

There are two types of pricing plans available in Snowflake for storage. The customers can choose the plan based on their use cases and needs. The best option is to start with the on-

demand and then switch to the pre-purchased capacity plan if your organization is new to Snowflake and not sure about the actual usage.

- **On-demand storage**: No Long-term licensing commitment. Pay for what you use month to month. This is like other cloud providers' pay-as-you-go model. A bill will be generated at the end of every month with the details for that month's usage. The cost for this plan is higher compared to the Capacity Storage plan.
- **Capacity Storage**: Long-term commitment is required with discounted pricing. Pay upfront in advance. Customers can purchase a set of Snowflake credits based on the pre-calculated capacity estimates. The cost for this plan is significantly lower compared to the on-demand plan.

Understanding the Snowflake cost model

Compute cost: The Compute cost is priced/billed on a per-second basis, with a minimum of 60 seconds whenever the warehouse starts, or resumes based on the hourly rate associated with the account. After 1 minute, all the subsequent billing is per second as long the warehouse runs continuously. Also, suspending and then resuming or resizing the warehouse will follow a similar minimum 60-second billing. Overall, the Compute Cost or Credits are calculated for the following activities related to any query or data processing to Snowflake.

For the virtual Warehouse, the total cost will be calculated based on the Warehouse size (X-Small to 6X Large), the number of virtual warehouses, and how long they run to complete the activity.

Following are the subsections of the Compute Cost Model. For additional details, refer below for what activities are included in each section:

- **Cloud Service Compute:** The Cloud Service layer performs various activities and services to tie together different components within Snowflake like user requests/login, query processing, query display, and much more. Cloud Service computing is completely managed by Snowflake.
- **Virtual Warehouse Compute:** The Virtual Warehouse consumes credits as they perform query execution, data loads, and perform any DML operation. This is managed by the user, so the users have the flexibility to control/manage how the virtual warehouse credits will be utilized to perform various operations within Snowflake.
- **Serverless Compute**: The serverless features are completely managed by Snowflake. Activities like Automatic Clustering, Snowpipe, Replication, Tasks, Materialized views, and so on, come under the Serverless compute category. Serverless compute cost is calculated based on the total usage of Snowflake managed to compute measured in compute hours. The compute hours are calculated on a per-second basis rounded up to the nearest whole second.

- **Storage cost**: The Snowflake Storage cost is based on the flat rate per **Terabyte** (**TB**), calculated monthly based on the average number of bytes stored on disk each day. This pricing can vary depending on the type of Snowflake account (On Demand or Capacity) and the cloud provider/region where the account is hosted on. Overall, the Storage Cost or Credits are calculated for the following activities related to data storage in Snowflake:
    - Data/File Storage for bulk data loading and unloading
    - Database Objects Storage: Tables, historical data for Time- Travel (1- 90 days)
    - Database Objects Storage: Fail Safe Storage cost (7 Days)
    - Clones of any database tables that reference data deleted in that table own the clones

Snowflake automatically compresses all the data stored in the database/tables and uses the compressed size to calculate the total usage for that account. It is not calculated based on the original file/data size that was received.

Snowflake vs. traditional architecture

Let us talk about how Snowflake is different from traditional architecture. **Traditional data warehouses** often utilize a *shared-disk* or *shared-nothing* architecture. This means data is physically divided and stored on one or more servers, each with its own processing power. While this can be effective for smaller datasets, it becomes cumbersome and expensive to scale as data volumes grow.

Snowflake, on the other hand, leverages unique hybrid architecture. Data is securely stored in a central cloud repository, separate from the virtual warehouses that perform computations. These virtual warehouses can be easily scaled up or down based on processing needs, offering significant flexibility and cost-efficiency. This separation offers several advantages:

- **Simplified management**: The centralized data storage eliminates the need to manage data across multiple servers, reducing complexity.
- **Elastic scalability**: Virtual warehouses can be scaled independently, allowing for efficient resource allocation and cost optimization.
- **Pay-as-you-go model**: Snowflake operates on a consumption-based pricing model, where users only pay for the resources they utilize.
- **Query performance**: Improves performance, as queries can be parallelized across multiple virtual warehouses for faster results.

In essence, Snowflake provides a more agile and cost-effective approach to data warehousing compared to traditional architectures. The following diagram illustrates how Snowflake architecture is different from the traditional Datawarehouse tools architecture:

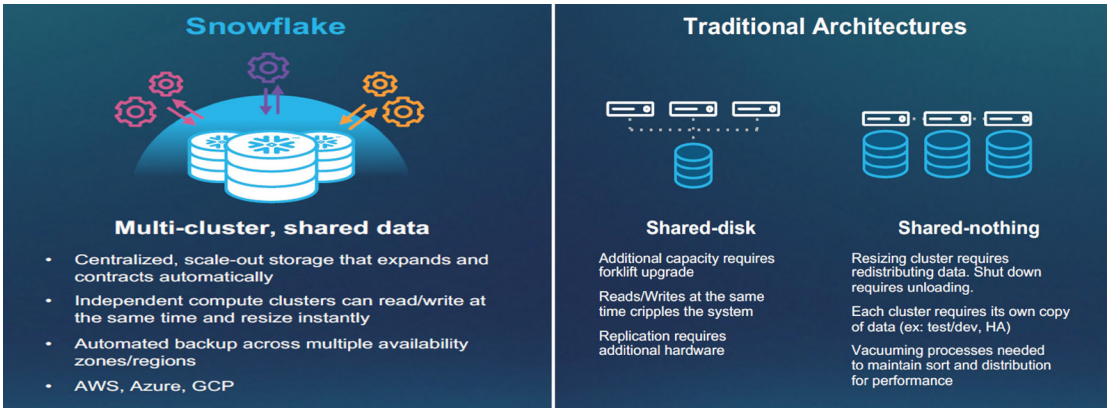

Figure 2.8: *Snowflake Vs Traditional Architecture*

Conclusion

In this chapter, we learned about the Snowflake cloud data platform, architecture, and technical details of the individual architecture components of the Service, Compute, and Storage layers. We also learned about the cloud providers, various editions, pricing details, and how Snowflake is different from other traditional data warehousing solutions.

In the next chapter, we will learn about the various unique key features specific to Snowflake, and information about data share, Partner Ecosystem, and various categories within the partner ecosystem of Snowflake.

Points to remember

Refer to the following summary for a quick reference to what we learned so far in this chapter:

What is Snowflake and its key features?

- Snowflake is a cloud-based new data platform designed to address the challenges of modern data warehousing. It is not built from any existing database or technology.
- Snowflake Cloud Data platform was founded by *Benoit Dageville*, and *Thierry Cruanes* in July 2012, later it was publicly launched after two years in October 2014.
- Snowflake is offered as a **software-as-a-service** (**SaaS**). Overall, Snowflake manages all aspects of hardware, software, upgrades, and configurations behind the scenes without any user interventions.
- The unique feature of the Snowflake architecture to the separation of storage and computing compared to other traditional data warehousing solutions. This new feature allows the organization/user to enable the following:

  - **Scalability**: Elastic virtual warehouses can be sized to meet specific workloads, ensuring optimal performance and cost-efficiency.
  - **Agility**: Automatic, frequent updates deliver new features and enhancements, keeping users at the forefront of data analytics capabilities.
  - **Ease of Use**: The cloud-native design eliminates the need for complex infrastructure management, simplifying data storage and analysis.
  - **Cost-Effectiveness**: The pay-as-you-go pricing model ensures users only pay for the resources they utilize.
- Snowflake supports Data Engineering, Data Lake, Data Science and ML, Data Applications, Data Exchange, and much more and empowers organizations to leverage their data for better decision-making through a robust, scalable, and user-friendly platform.
- Snowflake supports load and query the structured and semi-structured data in a single system.
- Snowflake new data type VARIANT used for loading semi-structured data from JSON, AVRO, Parquet, or ORC files.

The following are the types of cloud and non-cloud offerings:

- On-prem (non- cloud)
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

What are the key Architecture layers of Snowflake?

The Snowflake architecture build-up of the following three key layer architecture:

- Cloud Service or Service Layer
- Query Processing or Compute (Virtual Warehouse) Layer
- Database Storage or Storage Layer
- The fourth layer in Snowflake is the Cloud Agnostic layer is about which cloud provider the Snowflake account is hosted on. Snowflake was initially released in AWS and Azure platforms, later in 2020 GCP. Snowflake supports the following cloud providers based on customer choice.
  - Amazon Web Services (AWS)
  - Microsoft Azure
  - Google Cloud Platform (GCP)

The following are the list the available editions of Snowflake:

- Snowflake offers various editions to fit the organization's needs; following are the high-level:
    - **Standard edition**: Snowflake's introductory level offering
    - **Enterprise edition**: This is the second-level edition of Snowflake. Enterprise editions provide all features and services of Standard Edition Plus additional features specifically designed for large-scale enterprise organizations
    - **Business critical edition**: The next level to the Enterprise edition is Business Critical. This version is also formally known as **Enterprise for Sensitive Data** (**ESD**). Business Critical edition offers higher data protection than Standard/Enterprise to support organization that handles very sensitive data specifically PII and PHI data that require various regulations like HIPPA and HITRUST etc. This Edition includes all features and services of Enterprise Edition plus enhanced data protection and security.
    - **Virtual Private Snowflake (VPS)**: VPS Edition is the topmost offering that provides the highest level of security for any enterprise or organization that handles highly sensitive data, such as Banking, Financials, or similar large enterprise organizations that handle extremely sensitive data.

How Snowflake pricing model work?

- The Snowflake pricing is based on the volume of data that you store in Snowflake and the compute time.
- Snowflake uses a pay-as-you-go pricing model, meaning you only pay for the resources you actually use. Here is a breakdown of what they charge for:
    - **Storage**: You are charged a monthly fee based on the average amount of data you store in Snowflake after it has been compressed.
    - **Compute**: This is where most of the cost comes from. You pay per second for virtual warehouses that process your queries and load data. Warehouses come in different sizes, so you can choose the power you need for your workload. There is a minimum charge per warehouse start, but you only pay for the time it's actively working.
    - **Data transfer**: Snowflake charges a fee when you move data out of your Snowflake account to another region or cloud platform. There's no charge for bringing data into Snowflake.

How Snowflake compute cost is calculated?

- The Compute cost is priced/billed on a per-second basis, with a minimum of 60 seconds whenever the warehouse starts, or resumes based on the hourly rate associated with the account. After 1 minute, all the subsequent billing is per second as long the warehouse runs continuously.

- For the virtual Warehouse, the total cost will be calculated based on the Warehouse size (X-Small to 6X Large), the number of virtual warehouses, and how long they run to complete the activity.
- Compute Cost will be calculated based on associated workloads for Cloud, Virtual Warehouse and Serverless compute.

How Snowflake storage cost is calculated?

- The Snowflake Storage cost is based on the flat rate per **Terabyte** (**TB**), calculated monthly based on the average number of bytes stored on disk each day.
- This pricing can vary depending on the type of Snowflake account (on demand or capacity) and the cloud provider/region where the account is hosted. Following are the two types of pricing plans are available in Snowflake:
    - **On-demand pricing**: No long-term commitment, higher cost than the Capacity Storage plan. Pay on Month to Month
    - **Capacity Storage**: Long team commitment with discounted pricing. Pay upfront
- Snowflake automatically compresses all the data stored in the database/tables and uses the compressed size to calculate the total usage for that account.

How frequently does Snowflake perform releases?

Snowflake releases new features and updates frequently, typically every week. These updates happen in the background, so users will not experience any downtime and will always have the latest version. Here is a simplified breakdown of what you can expect in each type of release:

- **Regular releases (weekly)**: These releases introduce new features, enhancements, and bug fixes. They typically do not contain any breaking changes to how Snowflake works.
- **Behavior change releases (monthly)**: One release each month (except November and December) is chosen to potentially include changes that might affect how your existing Snowflake code works. These changes are usually minor, but it is a good idea to be aware of them.

Chapter 3
Snowflake Cloud Data Platform Features

Introduction

In the previous chapter, we learned about the Snowflake cloud data platform, architecture, and technical details of the individual architecture components of the service, compute, and storage layers. In this chapter, we will learn about the unique key features specific to Snowflake and how the Snowflake is different from other platforms. We also learn about the impact and benefit of each Snowflake's key features.

Structure

This chapter is going to cover the following topics:

- Key features overview
- Elastic storage
- Elastic compute
- Data cloud/data exchange
- Partner ecosystem
- Cloud partner categories

Objectives

This chapter aims to provide information about the key features of Snowflake and its unique architecture components such as elastic compute, storage, and additional components within the data cloud partner ecosystems.

Key features overview

As we learned in the previous chapters, Snowflake is a modern data warehousing solution built on the cloud from scratch. Snowflake, offered as a **Software-as-a-Service** (**SaaS**), aims to be a one-stop solution to support various modern data warehousing use cases like data engineering, data lake, data science and machine learning, data applications, data exchange, and much more.

Snowflake's unique elasticity supports any scale of data, processing, and workloads. This is the unique feature that the companies want to avoid setting up and maintaining multiple systems using other platforms for various use cases. No hassle to maintain and manage multiple systems or the hardware, software, or infrastructure and integrate all of them. No data redundancy or maintenance of multiple copies of data silos.

The following figure illustrates various modern data use cases around Snowflake:

Figure 3.1: *Snowflake modern data warehouse – use cases*

Let us call out a few unique key features of Snowflake:

- **Data warehouse as a service**: Snowflake is the first cloud data platform that offers data warehouse as a service, eliminating the need to deploy and manage hardware or software. This is an advanced platform delivered as a self-managed service. Snowflake enables the organization to build modern data warehousing solutions in the cloud (data storage, processing, and analytics) in a faster, easier, and more flexible style compared to traditional data warehousing solutions waiting for months to get ready.
- **Separation of compute and storage**: The unique key feature of the Snowflake architecture is the separation of storage and compute compared to other traditional data warehousing solutions. Technically, it is like the combination of shared disk and shared nothing database architecture.
- **Cloud offering**: Snowflake is the first cloud platform to support all three leading cloud providers: **Amazon Web Services** (**AWS**), **Microsoft Azure**, and **Google**

Cloud Platform (**GCP**). This feature provides complete flexibility for the customers to choose the cloud providers based on enterprise technology stack and needs. No other modern platform supports these kinds of flexibility. This is another versatile feature of picking Snowflake. The customers have the flexibility to choose one or more cloud providers based on their need to set up the Snowflake Account. The Snowflake offers across cloud providers and regions and is capable of delivering high throughput and low latency to workloads that need them.

- **All business data in one place**: Snowflake is the first platform that supports real-time data ingestion and transformation and stores various data formats like both structured and semi-structured data, and native SQL to query and analyze the data to find the insights in a matter of few minutes. Snowflake supports loading semi-structured data to a new column data type VARIANT. Users can directly load semi-structured data such as JSON, AVRO, Parquet, or ORC files without any additional transformation and query both structured and semi-structured data, even in a single query.
- **Multi-dimensional elastic cloud**: Snowflake's unique architecture enables it to support any scale of storage, computing, and users. The elastic cloud feature provides flexibility to create additional compute clusters to support as many workloads or users without moving or copying the data.
- **Time travel and fail-safe** are other key features for data recovery and backup to prevent data losses. No organization can afford to trade off the data losses. This is the *out-of-the-box* feature that allows the customers to recover or roll back the data for up to 90 days using the time travel option, plus seven days using Fail-safe. The time travel does not require Snowflake support. However, the Fail-safe is required to be working with Snowflake to retrieve the data beyond 90 days.
- **Pay only what you use**: The customers pay only for what they use, either on demand or pre-purchased capacity plans.
- **Snowflake editions**: Snowflake offers various editions to fit the organization's and users' specific needs. This enables customers to choose the editions that satisfy their requirements. The Snowflake editions start from standard to enterprise to business-critical to virtual private Snowflake.
- **Zero-copy cloning** is another amazing feature that gives the flexibility to clone the data at the point of time snapshots and share with a lower environment. No more DBA tickets and waiting for days/weeks to copy data from prod to lower environments. Faster development and delivery of product development.
- The **data cloud and data exchange** are another unique feature introduced by Snowflake that allows the organization to store and share data securely within or outside the organization. This feature empowers the customer to store data once and share it with multiple consumers without recreating the data copies.

- **Snowflake industry compliance**: Snowflake is certified by several industries-compliance like the **Health Insurance Portability and Accountability Act (HIPAA)**, SOC1 Type II, SOC2 Type II, PCI DSS, FEDRAMP, and HITRUST. These are a few, and Snowflake is continuously expanding this list.

These are just a few key features of Snowflake and much more to add. We will learn other available features in the upcoming chapters. The following figure illustrates how efficient snowflake elastic cloud differs from traditional Datawarehouse tools:

Figure 3.2: *Snowflake elastic cloud (Credit: Snowflake documentation)*

Elastic storage

The Snowflake's storage layer is also known as the **database storage layer**. When data gets loaded into Snowflake, the storage layer manages, organizes, and stores the data (structured, semi-structured, and unstructured) in its optimized, compressed format.

The data will be reorganized and stored in **columnar format** for faster retrieval. Snowflake handles all aspects of data storage. The data objects stored inside Snowflake are not directly accessible or visible to the customers. They are only accessible through SQL query operations using Snowflake.

Snowflake architecture to the separation of storage and computing features allows organizations to capture and process any scale of data, processing, and workload at blazing speed. This new feature allows the organization/user to scale compute on the fly and store an infinite amount of data instantly.

Snowflake automatically compresses all the data stored in the database/tables and uses the compressed size to calculate the total usage for that account. It is not calculated based on the original file/data size that was received.

The Snowflake storage cost is based on the flat rate per **Terabyte** (**TB**), calculated monthly based on the average number of bytes stored on disk each day. Overall, the storage cost or credits are calculated for the following activities related to data storage in Snowflake:

- Space occupied by Snowflake Internal Stage: Data/File storage for bulk data loading and unloading.
- Database and objects storage: Storage space for various Snowflake databases and objects like tables, stages, and other temporary storage cost.
- Data backup and recovery storage cost for time-travel (1- 90 days)and Fail-safe(7 days).
- Data share and data clone storage cost.

Elastic compute

The cloud service layer is also known as the **compute layer**. This layer manages and controls all the query execution using the **virtual warehouse**. The virtual warehouse relates to the compute, and this can scale up or down instantly at any time. We will learn more about virtual warehouses in the upcoming chapters.

Snowflake Elastic Computing is a cloud-based data warehouse service that allows you to scale your data storage and compute resources up or down automatically based on your needs. This means that you only pay for the resources you use, which can save lots of money on your data warehousing usage costs.

Snowflake Elastic Computing is built on a unique architecture that separates storage and compute. This means that you can scale your storage and compute resources independently of each other. For example, you can increase your storage capacity without increasing your compute power, or vice versa.

Snowflake Elastic Computing is also very easy to use. You can simply point your data at Snowflake, and it will automatically scale to meet your needs. There is no need to provision or manage any hardware or software.

Here are some of the benefits of using Snowflake Elastic Computing:

- **Cost savings**: You only pay for the resources that you use.
- **Scalability**: You can easily scale your data storage and compute resources up or down based on your needs.
- **Ease of use**: Snowflake is very easy to use and there is no need to provision or manage any hardware or software.
- **Performance**: Snowflake is a high-performance data warehouse that can handle even the most demanding workloads.

The following figure illustrates the Snowflake cost model for various serverless compute usage managed by Snowflake – pay only what you use:

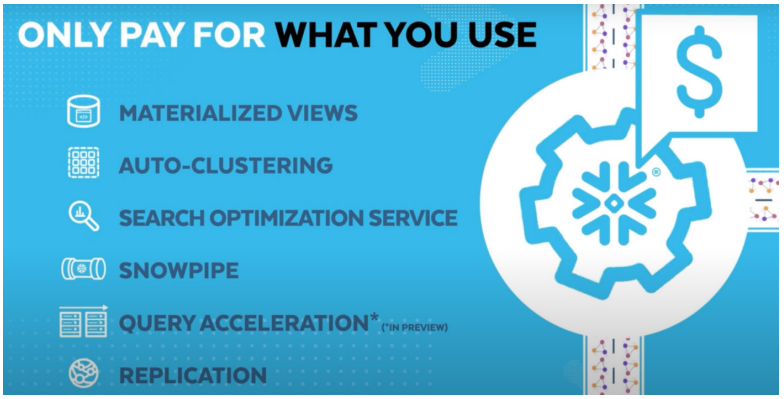

Figure 3.3: *Snowflake's compute cost model for Serverless Services (Credit: Snowflake documentation)*

Snowflake Serverless services automate common management tasks to meet your SLAs and organization budget. Unleash the turbo power of your data now and into the future with the Snowflake elastic performance engine.

Snowflake compute cost is priced/billed on a per-second basis, with a minimum of 60 seconds whenever the warehouse starts or resumes based on the hourly rate associated with the account. After one minute, all the subsequent billing is per second as long the warehouse runs continuously. Also, suspending and then resuming or resizing the warehouse will follow a similar minimum 60-second billing. Overall, the compute cost or credits are calculated for the following activities related to any query or data processing to Snowflake.

For the virtual warehouse, the total cost will be calculated based on the warehouse size (X-small to 6X large), the number of virtual warehouses, and how long they run to complete the activity. Following are the subsections of the compute cost model:

- **Cloud service compute**: It is completely managed by Snowflake, and the cost is involved for various cloud service layer activities.
- **Virtual warehouse compute**: This is managed by the user, so the users have the flexibility to control/manage how the virtual warehouse credits will be utilized to perform query execution, load data, and perform any DML operation. Snowflake's virtual warehouses offer on-demand elastic scaling through a combination of vertical and horizontal scaling mechanisms. This is different from traditional data warehouses where you're stuck with a fixed amount of power. Snowflake lets you scale your data processing power up or down dynamically and adjust compute resources based on workload requirements. **Vertical scaling** allows for increasing the processing power of individual warehouses, ideal for handling complex

queries. **Horizontal scaling** provides the ability to spin up additional warehouses, effectively distributing workloads across a larger pool of resources. This combined approach ensures optimal performance for varying workloads while optimizing costs by paying only for the utilized resources.

- **Serverless compute**: The Serverless features are completely managed by Snowflake. Serverless compute cost is calculated based on the total usage of Snowflake managed to compute measured in compute hours. The compute hours are calculated on a per-second basis and rounded up to the nearest whole second.

Data cloud and data exchange

The data cloud and data exchange are another unique feature introduced by Snowflake that allows the organization to securely store and share data within or outside the organization. This eliminates the data silos and no more complex ETL/ELT by sending the files between one or more internal or external entities.

The secure data share lets the organization share one or more selected objects in your account with other Snowflake accounts. This can be within the organization or outside accounts. This is the key unique feature of Snowflake. Load/Store the data once and share as many.

Imagine you have valuable data but want others to analyze it without giving away the actual information. Snowflake's secure data sharing feature lets you do just that! Following is the quick summary of Snowflake data share.

Snowflake supports the following data sharing technologies to share with other Snowflake accounts:

- **Direct data sharing**: In a direct data share, the Snowflake customer directly shares a specific database object (data share) with another account in the same region and the cloud provider.
- **Data or private exchanges** are used to set up and manage a group of accounts and offer a share to that group. This is useful for sharing data with a large number of accounts or with accounts that are not in the same region as your account.
- **Snowflake data marketplace**: The Snowflake data marketplace enables customers to discover, evaluate, and purchase the data, data services, and applications the organization needs to innovate for their business, giving direct access to third-party products with some of the world's leading data and solution providers. The Snowflake customers can provide and consume various data listings offered publicly or privately using the Snowflake marketplace.

The following figure illustrates different data sharing technology types:

Figure 3.4: *Snowflake data sharing types*
(Credit: Snowflake documentation)

Snowflake Marketplace offers a variety of data sources from the marketplace listings. Users can find the following free, free-to-try, and paid listings under the most popular categories. Refer to the following figures for a few sample listings from the Snowflake marketplace:

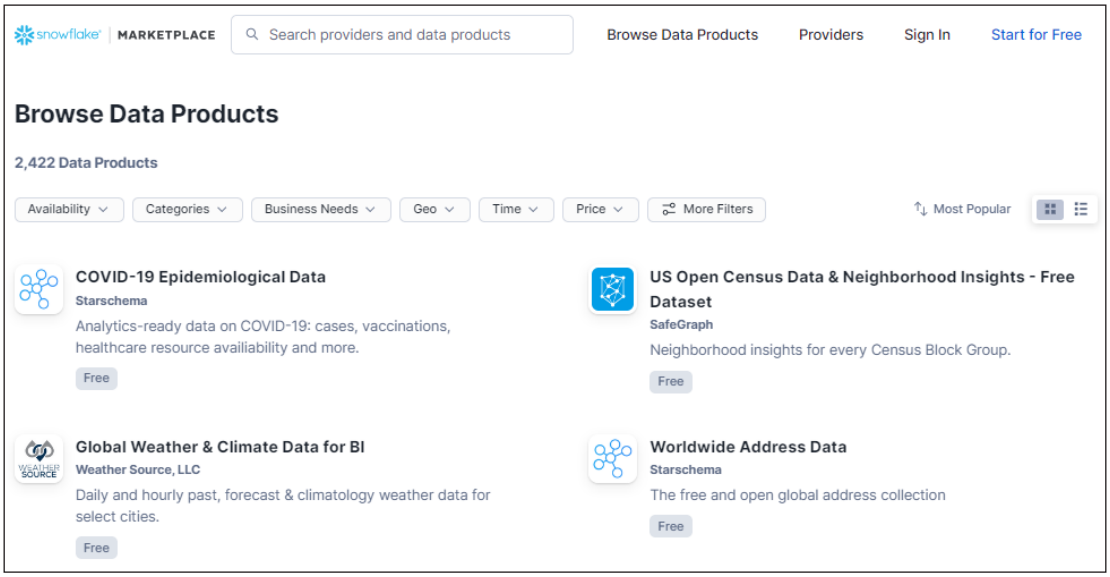

Figure 3.5: *Snowflake marketplace – most popular listings*

Users can find the following Free, Free-to-try, and Paid listings under the most recent categories. Refer to the following figures for a few most recent sample listings:

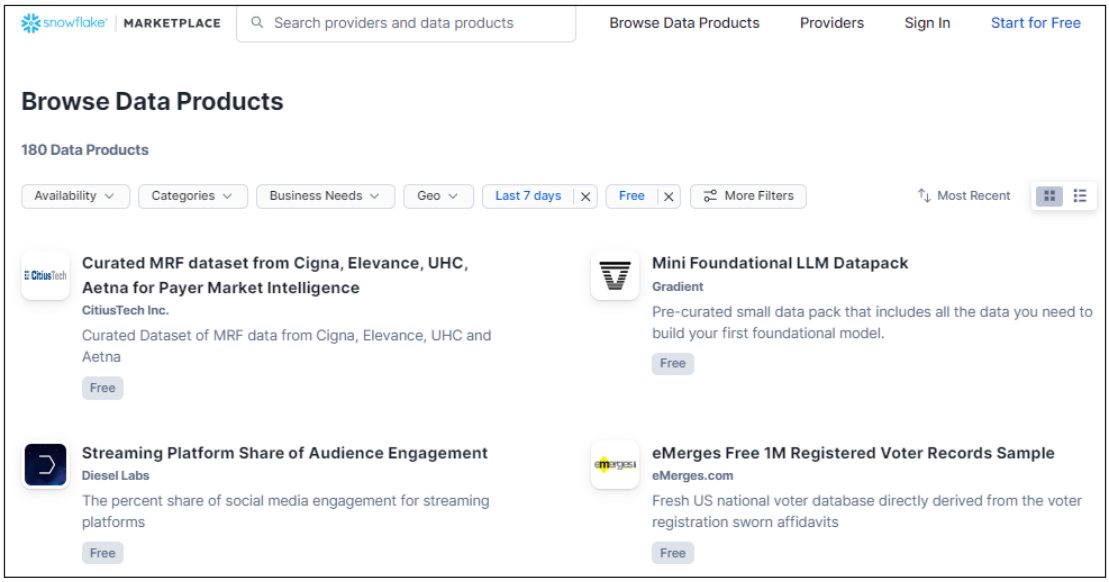

Figure 3.6: *Snowflake marketplace – most recent listings*

Snowflake marketplace: Domain-specific listings

In addition to the most popular, most recent listings, users can also find the listings by various most frequently used domain-specific key datasets. For example, refer to the following figures for a few financial domain listings from the Snowflake marketplace:

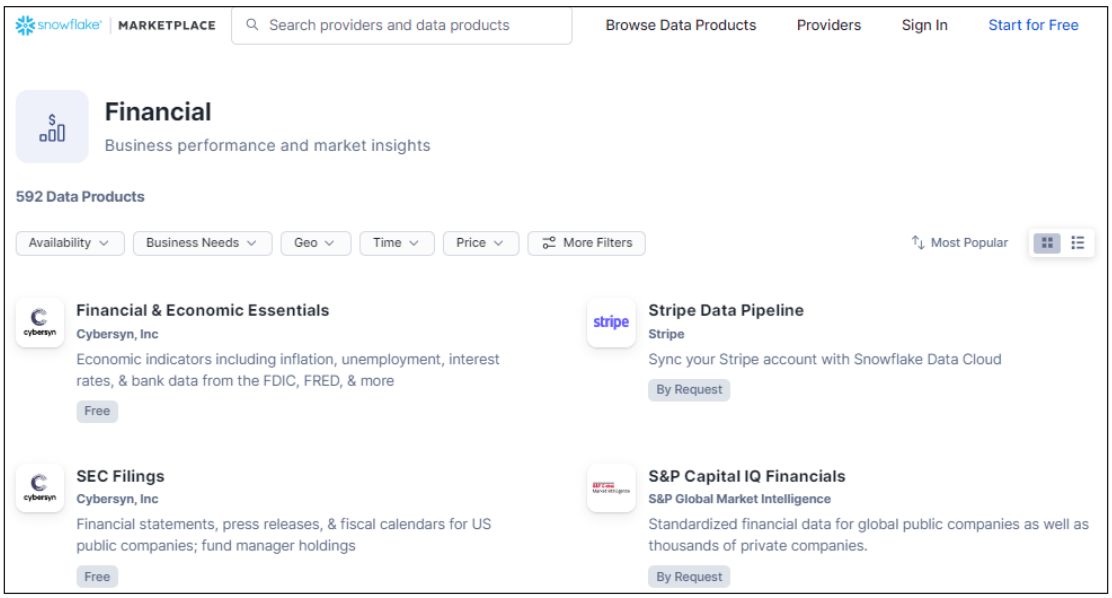

Figure 3.7: *Snowflake marketplace- financial listings*

Users can find the following free, free-to-try, and paid commerce domain listings such as consumer purchase and intent insights. Refer to the following figures for a few sample listings:

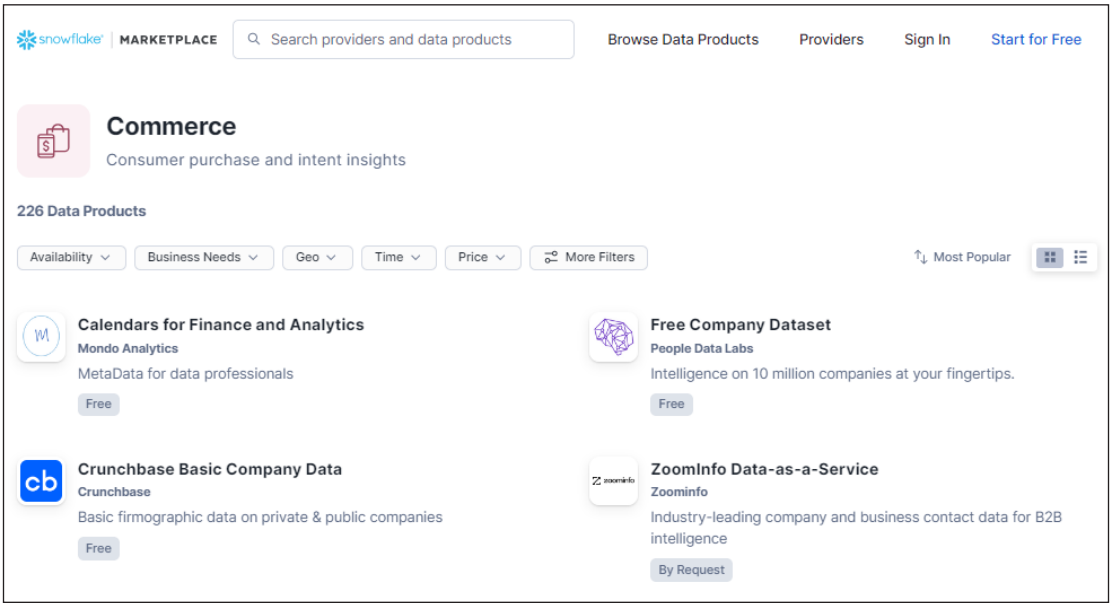

Figure 3.8: *Snowflake marketplace – commerce listings*

Users can find the following free, free-to-try, and paid demographics domain listings, such as socioeconomic insights about individuals and households. Refer to the following figures for a few sample listings:

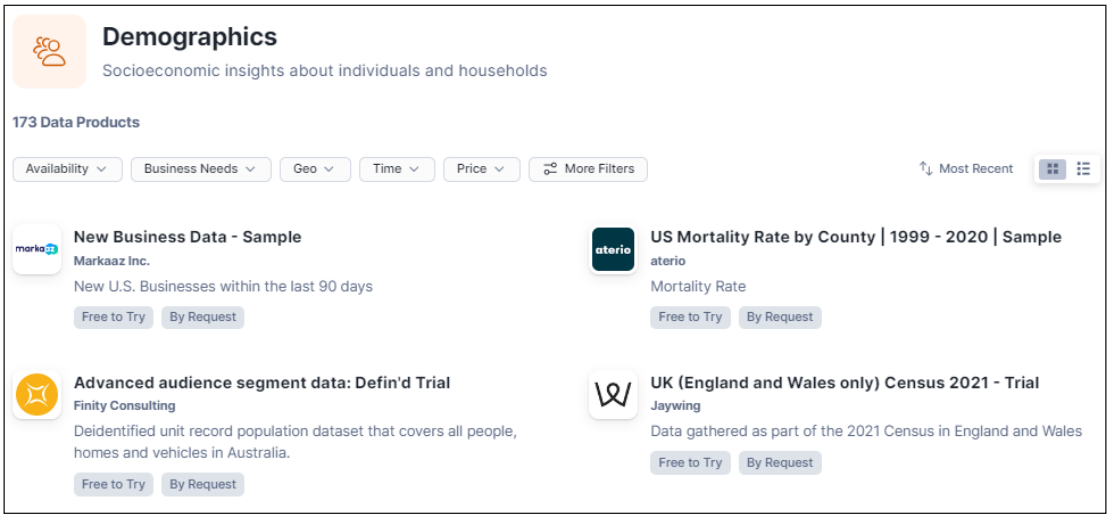

Figure 3.9: *Snowflake marketplace – demographics listings*

Users can find the following free, free-to-try, and paid weather domain listings such as Weather and climate of a region, and so on. Refer to the following figures for a few sample listings:

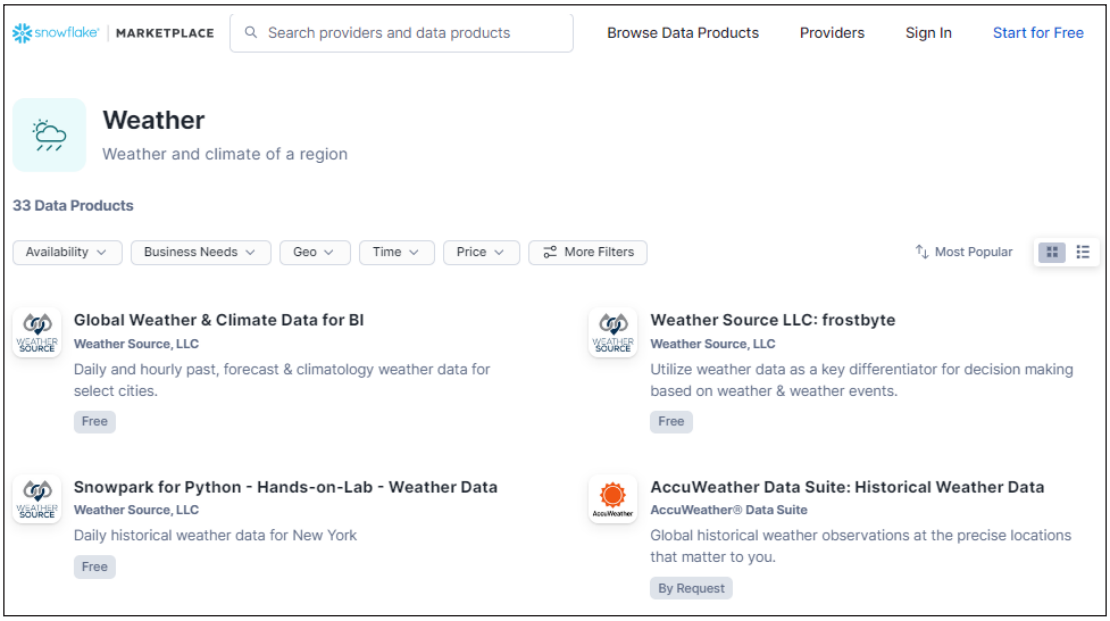

Figure 3.10: *Snowflake marketplace – Weather domain listings*

Users can find the following free, free-to-try, and paid health and life sciences domain listings such as public health, physician, provider, and patient insights. Refer to the following figures for a few sample listings:

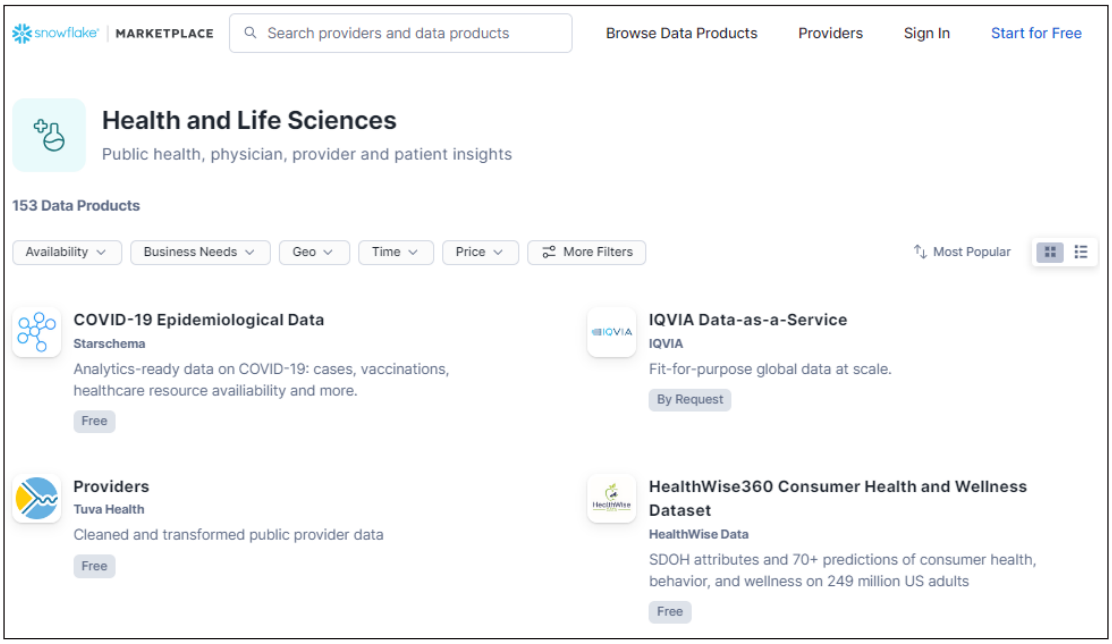

Figure 3.11: *Snowflake marketplace – health and life science listings*

Partner ecosystem

Snowflake provides several methods to interact with Snowflake. The Partner ecosystems consist of various third-party tools and technologies and Snowflake-provided clients make this connection seamless and work in a few click integrations with Snowflake.

The Snowflake enables customers to access Snowflake through an extensive network of a wide variety of industry-leading tools and technologies, drivers, connectors, utilities, and programming languages, including:

- Snowflake provided client tools such as SnowSQL (command line)
- Snowflake connectors for Python and Spark
- Snowflake drivers for ODBC, JDBC, Node.js, and much more.
- Cloud-based and on-premises solutions developed by certified partners for connecting to Snowflake.
- Other third-party tools and technologies are known to work with Snowflake.

Let us continue to the next section to learn more about the partner ecosystem along with a few examples.

The following figure illustrates the high-level list of tools, technologies, drivers, connectors, and so on, available in the partner ecosystem:

Figure 3.12: *Snowflake – ecosystems tools and technologies (Credit: Snowflake documentation)*

Cloud partner categories

The Snowflake customers can unlock the full potential of the data cloud with a broad array of tools, and technologies by leveraging trusted partners to deliver more meaningful data insights. Following are the various types of partner network categories:

- **Cloud partners**: Snowflake is a multi-cloud technology uniquely built to be global and hosted on one or more cloud environments. Customers can choose the cloud provider and the region-specific to their requirements regardless of the region or cloud providers.

  Supported cloud partners: AWS, Azure, and Google Cloud

- **Service partners**: The Snowflake customers can unlock the full potential of Snowflake by working with trusted Consulting Service Partners to deliver

industry-focused customized solutions. With this trusted relationship, customers can mitigate risk, well align organizations' goals and objectives with smooth transitions to take the business to the next level.

The service partners are further categorized into different program tiers such as Elite, Premier, and select based on the number of implementations and their number of SnowPro-certified consultants.

Refer to the following section for a few service partners' examples:

- **Elite service partners**: Accenture, LTIMindtree, Cognizant, ET, and many more
- **Premier service partners**: AHEAD, Analytics8, Genpact, and many more
- **Select service partners**: IBM, Servian, Apexon, Tech Mahindra US, and many more

Technology partners: Snowflake's Technology partnership empowers customers to enable flexibility, performance, and ease of use and implement Snowflake. The technology partners are part of the Snowflake Data cloud that natively connects to other solutions within the ecosystems, so the customers can easily deliver more meaningful data insights without any impact, risk, or additional cost to integrate various data sources from legacy systems.

The technology partners are further categorized based on implementation support or usage type with Snowflake.

Refer to the following list for technology partner categories, along with a few examples:

- **Data integration**: dbt Labs, Fivetran, Matillion, Talend, Qlik, and many more
- **Business intelligence (BI)**: SAP, ThoughtSpot, Tableau, Sigma, and many more
- **Machine learning and data science**: Alteryx, Dataiku, DataRobot, and many more
- **Security, governance:** Protegrity, SecuPi, Satori Cyber Ltd., Acceldatainc, Anodot, DQLabs, and many more
- **SQL development and management:** SnowSQL, Snowsight, SqlDBM, Aginity, APOS Systems, Data Gaps Inc., and many more
- **Native programmatic interfaces**: ODBC, JDBC, Python, GO, Node.js, Microsoft .NET, PHP, and so on.

Conclusion

In this chapter, we learned about the various unique key features specific to Snowflake, such as the separation of compute and storage, elastic compute and storage, data cloud and exchange, Partner Ecosystem, and various categories within the partner ecosystem of Snowflake. With all these features, now we can easily understand how Snowflake differs from other platforms.

In the next chapter, we will learn the step-by-step instructions on how to sign up for a Snowflake trial account, the Snowflake UI, how to navigate within UI and other available features like Snowsight, and the list of various drivers, and connectors to interact with Snowflake.

Points to remember

Refer to the below summary for a quick reference to what we learned so far in this chapter:

- List a few snowflake's key features:

  The Snowflake is a modern data warehousing solution built on the cloud from scratch. Snowflake's few specific key features are:

  - Data warehouse as a service
  - Separation of compute and storage
  - Multi-dimensional elastic cloud
  - Multi-cloud offering
  - Time travel and failsafe
  - Zero-copy Cloning
  - The data cloud and data exchange

- Describe Snowflake Storage and Compute layers:
  - The Snowflake's Storage layer is also known as the **database storage layer**. When data gets loaded into Snowflake, the Storage layer manages, organizes, and stores the data in its optimized, compressed format.
  - The Compute layer is also known as the Query Processing Layer. The Compute layer manages and controls all the query execution using the *Virtual Warehouse*. The virtual warehouse relates to the compute and this can scale up or down instantly at any time.

- List various types of data sharing methods available in Snowflake:

  Snowflake supports the following data sharing technologies to share data with other Snowflake accounts:

  - Direct data sharing
  - Data or private exchanges
  - Snowflake data marketplace

- List the types of Snowflake cloud partner categories:
  - Cloud partners
  - Service partners
  - Technology partners

- List of technology partner categories, along with a few examples (quick reference):
    - **Data integration**: dbt Labs, Fivetran, Matillion, Talend, Qlik, and many more.
    - **Business Intelligence (BI)**: SAP, ThoughtSpot, Tableau, Sigma, and many more.
    - **Machine learning and data science**: Alteryx, Dataiku, DataRobot, and many more.
    - **Security, and governance**: Protegrity, SecuPi, Satori Cyber Ltd., Acceldatainc, Anodot, DQLabs, and many more.
    - **SQL development and management**: SnowSQL, Snowsight, SqlDBM, Aginity, APOS Systems, Data Gaps Inc., and many more
    - **Native programmatic interfaces**: ODBC, JDBC, Python, GO, Node.js, Microsoft .NET, PHP, and so on.

Chapter 4
Snowflake Tools and User Interfaces

Introduction

In this chapter, we will learn the step-by-step instructions on how to sign up for Snowflake in a few clicks, and various available options to select during the free trial account sign-up process. Once we establish the trial account, the next steps are to learn about the Snowflake **user interface** (**UI**), how to navigate within UI and other available features like Snowsight, and the list of various drivers, and connectors to connect Snowflake.

Structure

In this chapter, we will go through the following topics:

- Getting started with Snowflake
- Snowflake user interfaces
- Snowsight
- Snowflake connectors and drivers
- Snowpark

Objectives

This chapter aims to help the readers achieve the first step of the Snowflake milestone to sign up for the trial account with their choice of Snowflake edition, cloud provider, etc. Users will gain knowledge beginning from Snowflake account setup to various available features within Snowflake to try, explore, learn, and validate various use cases utilizing the trial account setup.

Getting started with Snowflake

This is the simplest process; anyone can sign up and spin the Snowflake in a few clicks. No more weeks of planning and months to implement the data warehouse system. Snowflake is offered as **Software-as-a-Service** (**SaaS**). Overall, Snowflake manages all aspects of hardware, software, upgrades, and configurations behind the scenes without any user interventions.

Snowflake offers all three major cloud providers to choose from various editions and features. So, the customers have complete flexibility to choose one or more cloud providers based on their organization's need to set up the Snowflake Account.

How-to sign-up Snowflake

Follow the step-by-step instructions below to sign up for Snowflake. The trial account comes with a 30-day free trial that includes $400 worth of Snowflake usage to try, explore, and validate various use cases.

1. Fill in the basic necessary information like name, email, and so on, then click the **Continue** button:

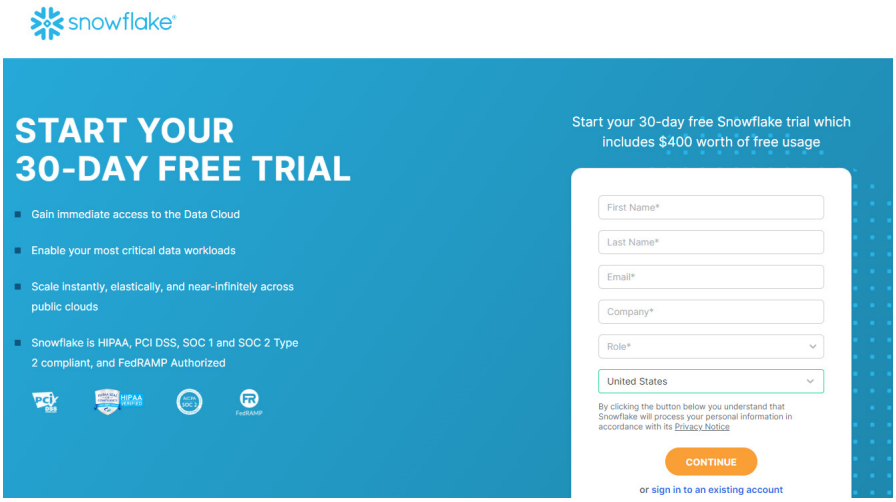

Figure 4.1: *Signup Snowflake– provide basic information*

Once the initial information section is complete in step 1, continue to the next page to fill in further details.

2. Select the Snowflake edition, cloud provider, and region that fits your needs and click on **Get Started** button:

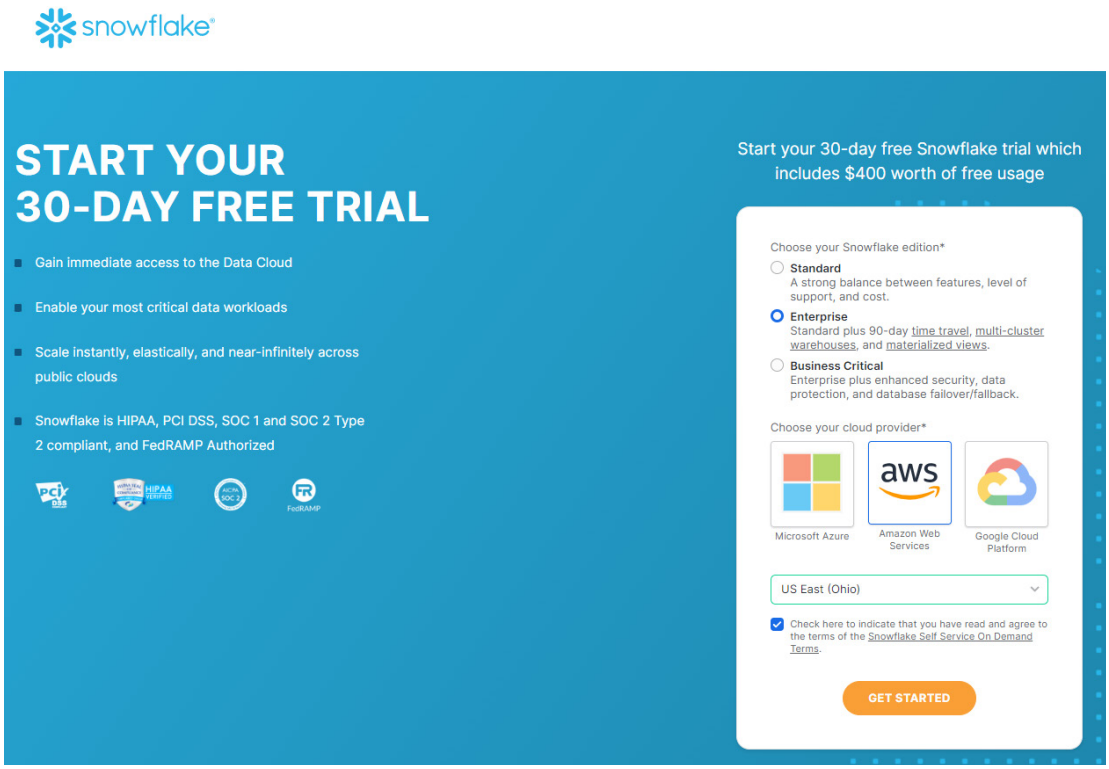

Figure 4.2: *Signup Snowflake – choose Snowflake edition and cloud provider*

Once the Snowflake edition, Cloud Provider, and Cloud Provider region selection is complete in step 2, continue to the next page to fill in further details.

3. This is optional. Choose **Skip** or select one or more appropriate available list of options for **Why are you signing up for a trial?**.

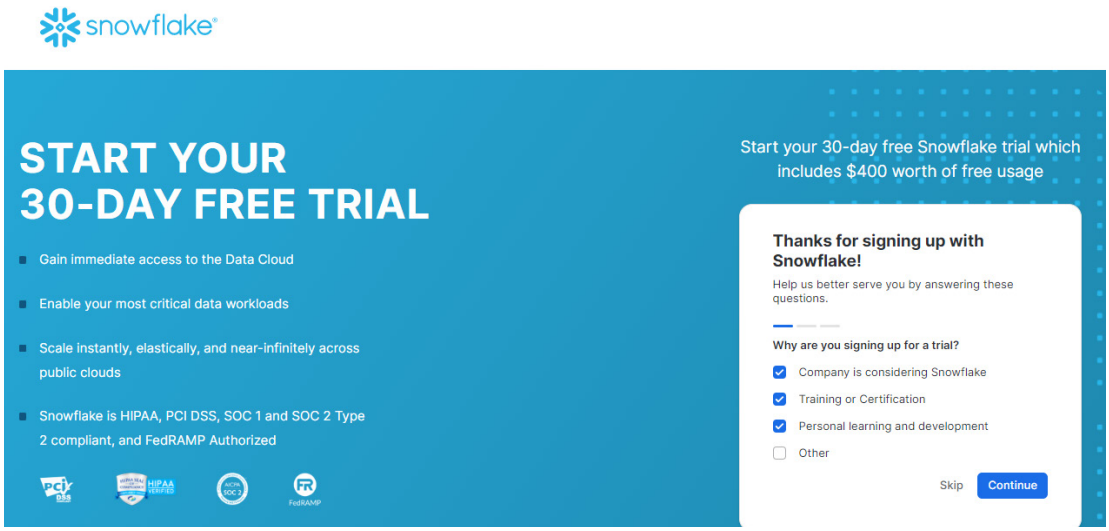

Figure 4.3: *Signup Snowflake – choose the purpose/reasons*

Once step3 is complete, continue to the next page to fill in further details.

4. This is optional. Choose **Skip** or select one or more appropriate available list of options for **What will you use Snowflake for?**

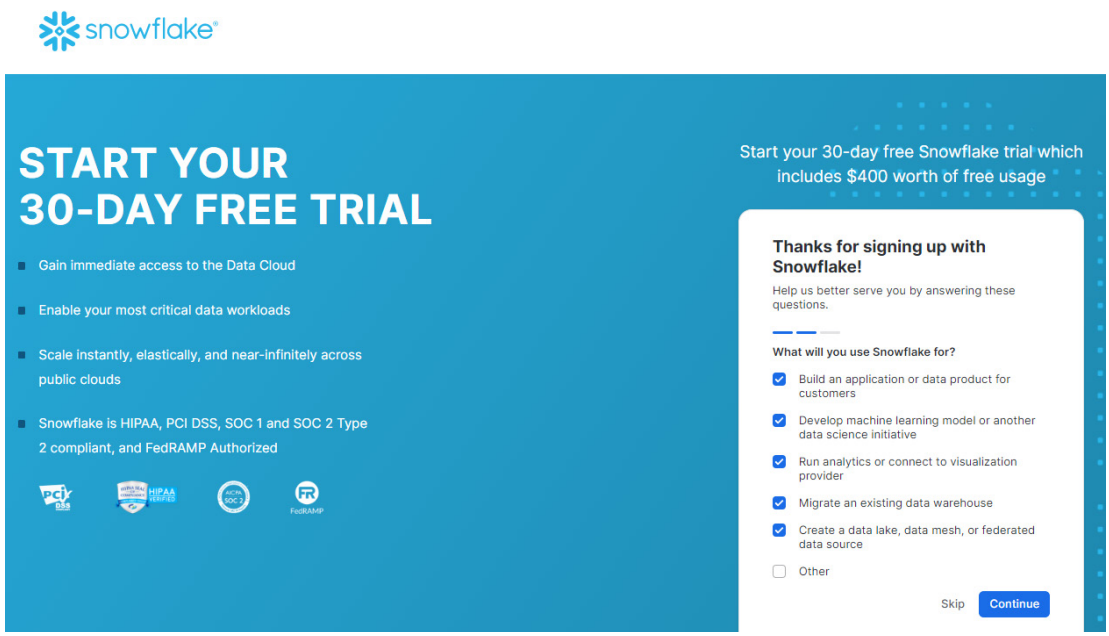

Figure 4.4: *Signup Snowflake – choose the use cases*

Once step 4 is complete, continue to the next page to fill in further details.

5. This is optional. Choose **Skip** or select one or more appropriate available list of options for **Select your preferred languages(s) to work in**:

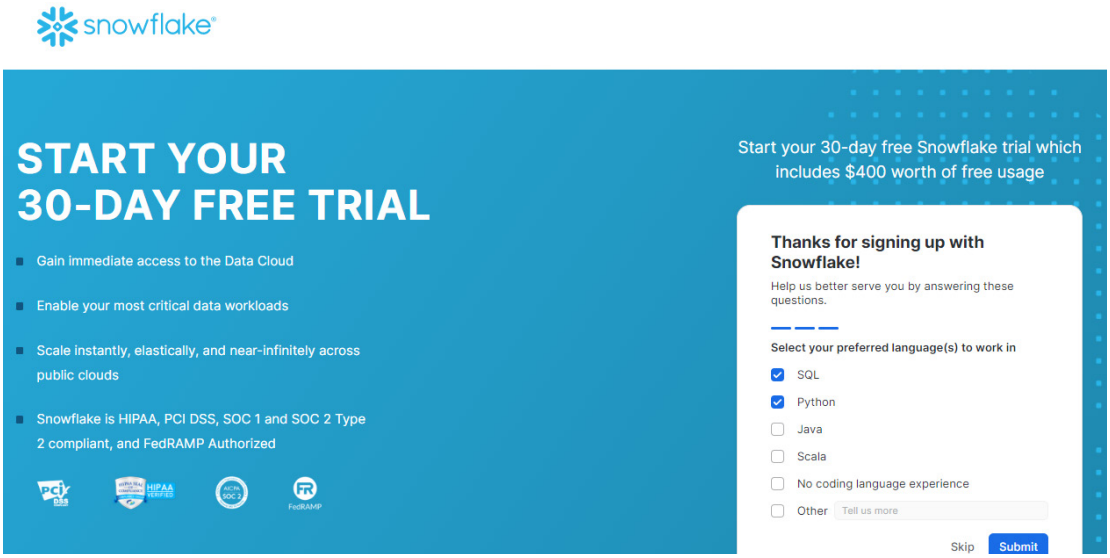

Figure 4.5: *Signup Snowflake – choose the preferred programming languages*

Once the optional selections to choose the use cases, and preferred programming languages(s) to sign up for the Snowflake account, continue to the last step to activate your account.

6. This is the final step; you have already provided all the necessary information to signup Snowflake. In this step, check your email to activate the Snowflake Account:

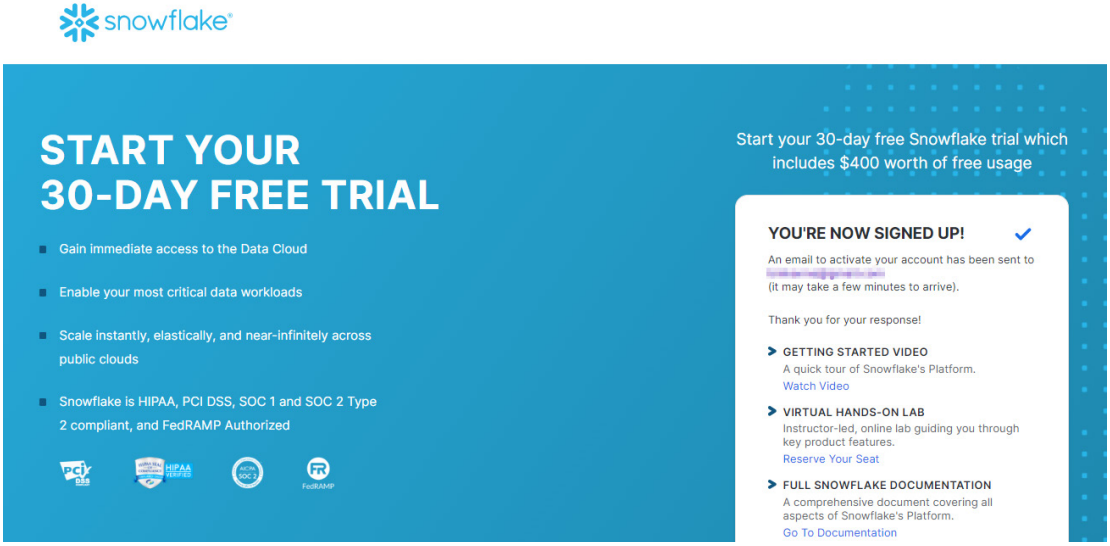

Figure 4.6: *Snowflake Signup – Check email and Activate Account*

7. Sample email content to activate the account:

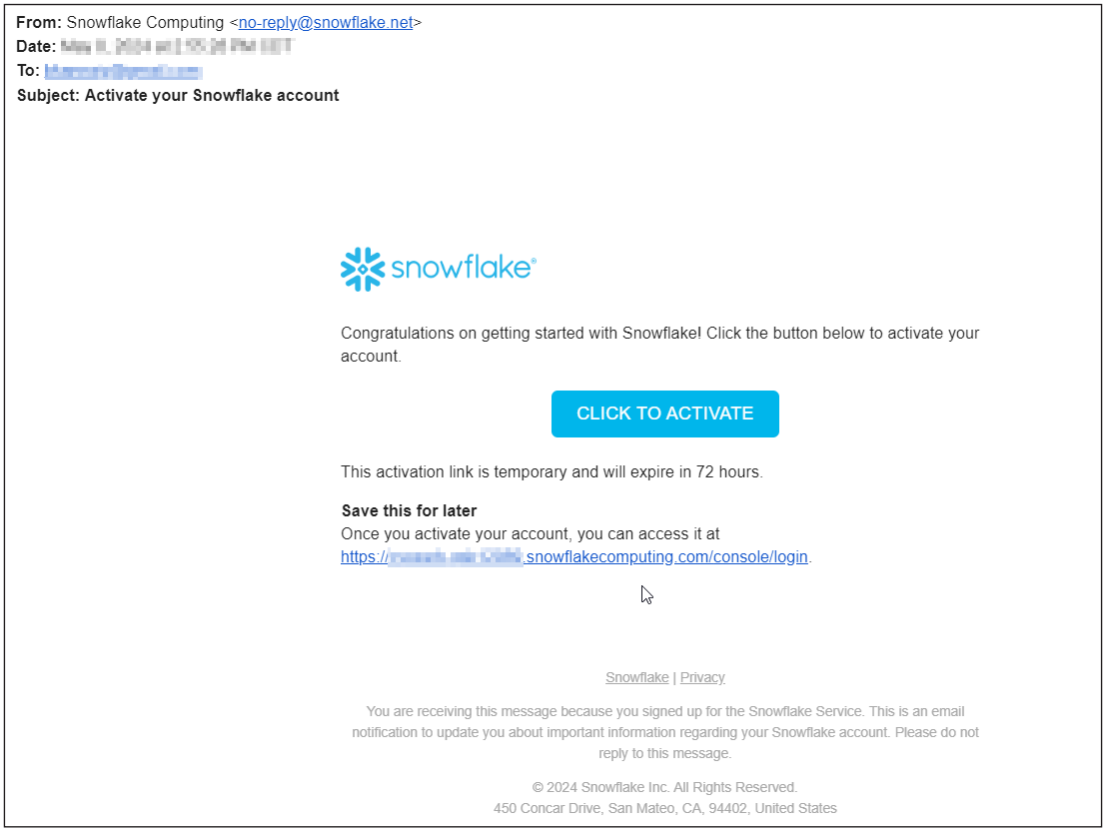

Figure 4.7: *Snowflake Signup – Check email and Activate Account*

Congratulations! Your Snowflake account is active. This 30-day free trial includes $400 worth of Snowflake usage to try, explore, and validate various use cases. Set the user ID and password part of this first-time activation process.

Snowflake user interfaces

Snowflake provides various **user interfaces (UI)** to connect and interact with the modern data platform. The following are the main user interfaces for Snowflake:

- Snowsight: The Snowflake Web Interface
- Classic Console(Note: Some accounts only have access to Snowsight, and no longer have access to Classic Console. For more details, refer to the link **https://docs.snowflake.com/en/user-guide/ui-snowsight-upgrade-gui**de)
- SnowSQL (CLI Client)
- Snowflake Extension for Visual Studio Code

- General Configuration (All Clients)
- Third-party tools and technologies

Let us discuss the next section to learn more about the key Snowflake interfaces.

Snowsight

Snowsight is Snowflake's easy-to-use web interface that enables users to interact and perform various operations with the powerful cloud data platform. Using Snowsight, users can perform the following operations:

- Primary UI to access/manage various activities within Snowflake
- Using Snowsight to manage your account, user profile, user preferences, change passwords
- To enable multi-factor authentication
- Develop, debug, and run various queries
- Create and modify the databases and all the database objects
- Create and manage users and account-level objects
- Monitor query performance and history
- Create, use, and monitor the virtual warehouses
- Create and share with other Snowflake accounts
- Explore various available public data sets/listings from Snowflake Marketplace
- User profile, preferences setup, and update

Getting started with Snowsight

This section describes how to get started with Snowsight. The Snowsight is the most used interface of Snowflake. Snowflake users can access Snowsight over the Internet to connect to the DataCloud. The Snowsight UI supports the following major versions of internet browsers:

Browser requirements:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Apple Safari

How to sign in to Snowsight

Follow the steps to access Snowsight over the public Internet:

1. Use one of the Snowsight Supported web browsers and navigate to the URL: **https://app.snowflake.com**, refer to the following figure:

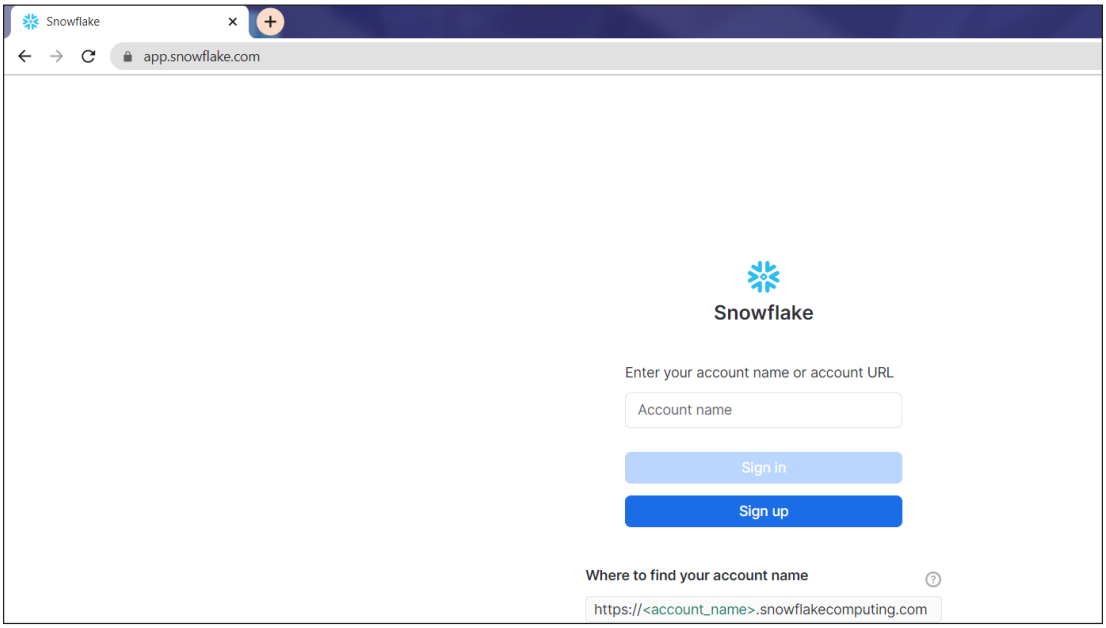

Figure 4.8: *Snowflake - Log-in Page*

2. Provide your account name and credentials to sign into your Snowflake account. The Account name can be found on the Snowflake signup activation link.
3. Users have the flexibility to access Snowflake either directly login to Snowsight or from Snowflake Classic Console. If you are already logged in to Snowflake Classic Console, then select Snowsight from the navigation menu to switch from Classic to Snowsight. Snowsight opens in a new tab.

Note: Some accounts only have access to Snowsight, and no longer have access to Classic Console.

Sign into a different Snowflake account

In this section, we will discuss how to sign into different Snowflake accounts from one to another. At the bottom of the left navigation page in Snowsight, use the account selector to sign into a different account. For user convenience, this option shows all the previously accessed accounts.

Using Snowsight: Various sections inside Snowsight

This section provides an overview of how to navigate various sections inside Snowsight UI:

- **Account selector**: The account selector, located at the bottom of the left nav, lets you sign into other Snowflake accounts. For user convenience, the account selector shows the list of accounts that the user signed in previously.
- **User menu**: The user menu, located at the top of the left nav, lets you:
    - Select from one or more available roles, such as **`ACCOUNTADMIN`**, **`SYSADMIN`**, and so on.
    - Switch from one Snowflake role to another role
    - View profile and current preference setup
    - Modify user preferences like changing passwords, updating name, and email address, and so on.
    - Access partner connect
    - Open/access support cases
    - Sign out Snowflake Web UI
    - Snowflake documentation
- **Exploring the Snowsight navigation menus**: The Snowsight UI layout changes from time to time as part of the Snowflake new feature releases. Refer to Snowflake documentation for the latest features and navigation changes in Snowsight | **https://docs.snowflake.com/en/user-guide/ui-snowsight-navigation**
- **Overview of the Snowsight New Navigation:**

  The Snowsight menu is reorganized! Instead of a single list of options, things are now grouped based on functionalities. Also, some features that were previously bundled together now have their own dedicated menu items for easier access.

  The following figure illustrates a new grouping and menu items of the latest Snowsight UI:

New grouping	Menu items contained within
Projects	**Worksheets**, **Streamlit**, **Dashboards**, **App Packages**
Data	**Databases**
Data Products	**Marketplace**, **Apps**, **Private Sharing**, **Provider Studio**, and **Partner Connect**.
Monitoring	**Query History**, **Copy History**, **Task History**, **Dynamic Tables**, and **Governance**.
Admin	**Warehouses**, **Cost Management**, **Users & Roles**, **Accounts**, **Security**, **Contacts**, and **Billing & Terms**
Account menu	Switch your active role, sign in with another account, access your **Profile**, **Support**, and links to **Documentation**. You can also access your account URL and account identifier.

Figure 4.9: *Snowflake– Snowsight's new grouping and menu items (Source: Snowflake documentation)*

The following figure illustrates a sample Snowsight User Interface - **Admin** section. This gives a sneak peek of the Snowsight:

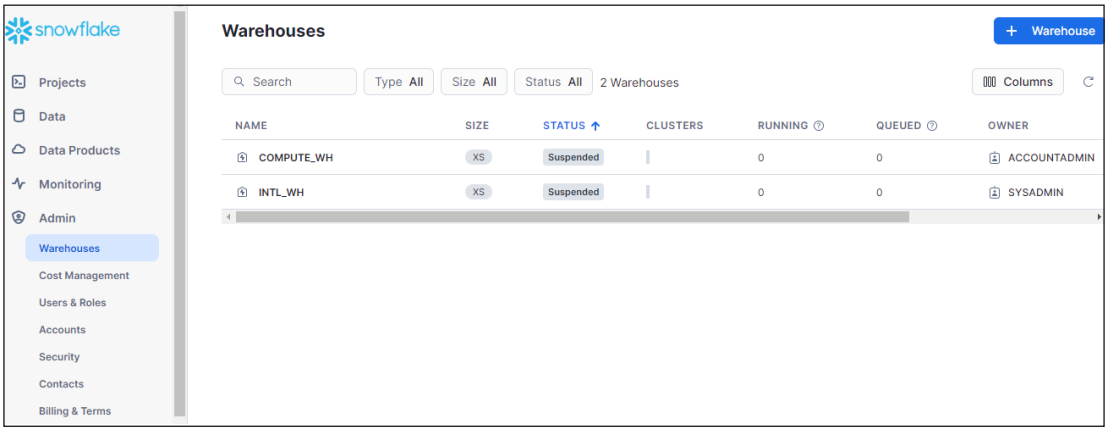

Figure 4.10: *Snowflake– Snowsight User Interface Admin Section*

Setting user details and preferences

From the user menu, select Profile to access your user profile section and details. In this section, the user can view and edit the following user details.

Please make sure the user profile has all the necessary information like first name, last name, and email address, as it is required for Snowflake to validate these details for accepting terms of service related to the Snowflake Marketplace. For example, the following are the available user preferences under the Snowflake Profile section:

- Username (cannot be changed)
- First name
- Last name
- Profile photo
- Password
- Email

Now let's see what this profile section looks like in the Snowflake UI. The following figure shows the available user preference under the **Profile** section:

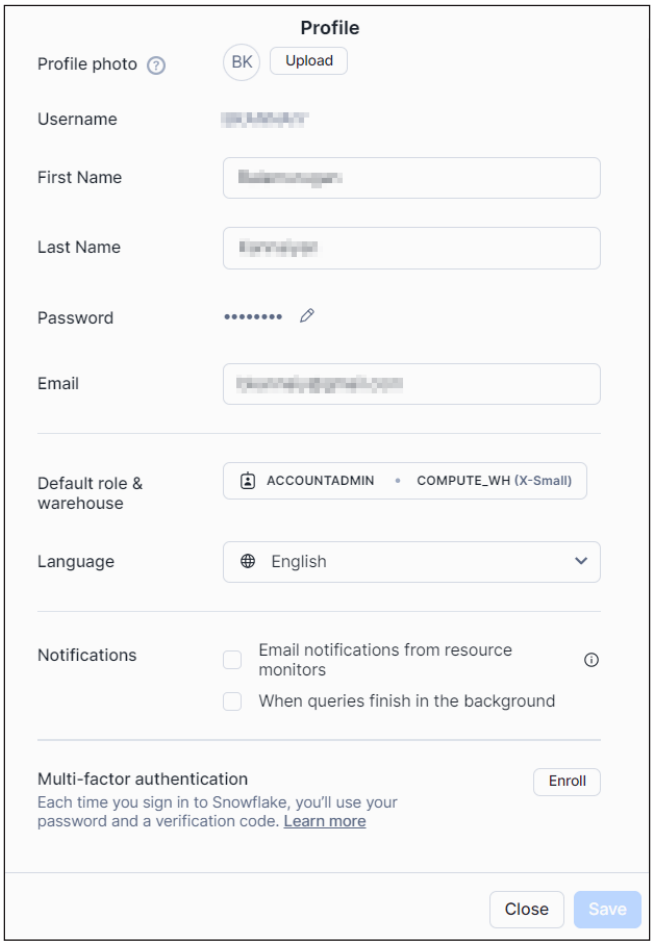

Figure 4.11: *Snowsight – Profile section*

Click close or Save to reflect any changes that the user made to the Profile section of Snowflake. The profile section can be changed anytime based on user preference.

Once the basic profile setup is done, there are a few optional setups available to the users to set the preference based on the day-to-day usage of Snowflake UI like default Role, Warehouse, etc. Refer to the next section for further details.

Additional user preferences

The following are the additional user preferences:

- **Default role and warehouse**: Users can set up the default role and warehouse from the available role/warehouse assigned for the user. This helps save time to set the user's most frequently used roles/warehouse that they want to use by default when they login to Snowflake every time.

- **Default experience**: Select which Snowflake Console you want your default web interface to interact with Snowflake. Users can set the default UI to Snowsight or Snowflake's Classic Console. Based on this selection, the default web interface will appear every time the user logs in to Snowflake. Note this option is available only to the older accounts and not available to all accounts.
- **Language**: Snowflake currently supports the following languages, choose from the selection:
    - English (US)
    - Japanese
- **Notifications**: There are two notifications available to enable as part of the user profile. The first one is Email notifications from the resource monitor (Available only to Account Admin Role). This option enables the notification for resource monitor for one or more warehouses. The second one is to set the notification for Notify when queries finish in the background (available to all the roles). This option enables the notification for background queries.
- **Multi-factor authentication**: Snowflake's **multi-factor authentication (MFA)**is powered by the Duo Security service. Enable based on the preference for second-level authentication/security.
- **Switching one active role to another**: To switch your active role, follow the mentioned steps:
    - Select the user menu from at the top of the left navigation section
    - Hover over the current active role.
    - Select/change the role that you want to use from the available roles assigned to the user.
- **Snowsight versus Classic Console**

  Snowsight and Snowflake Classic Console are two different UI interfaces that perform similar operations. However, there are a few minor differences between the consoles as follows:

    - Snowflake Worksheets: The worksheets are not synced between Classic Console and Snowsight. If the user makes query updates in the Classic Console worksheet, that will not be reflected in the Snowsight worksheet, and vice versa.
    - Usage information may be different in Classic Console due to it having a 2 million rows query limitation to calculate usage. This differs in Snowsight, so it is recommended to use Snowsight for accurate usage information.

The following figure illustrates Snowflake Snowsight and different sections within the UI:

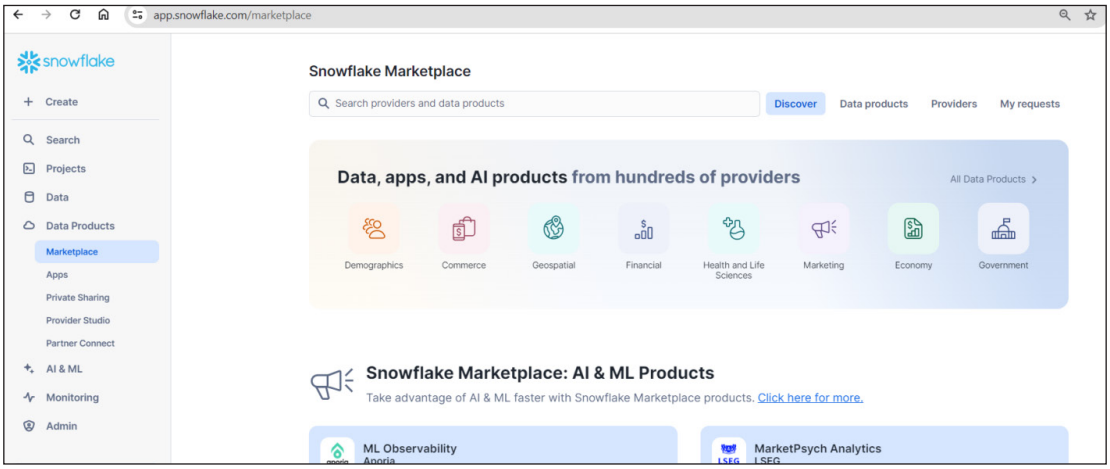

Figure 4.12: *Snowflake – Snowsight User Interface*

The following figure illustrates Snowflake Classic Console and different sections within the UI (Note: Available only to the older accounts):

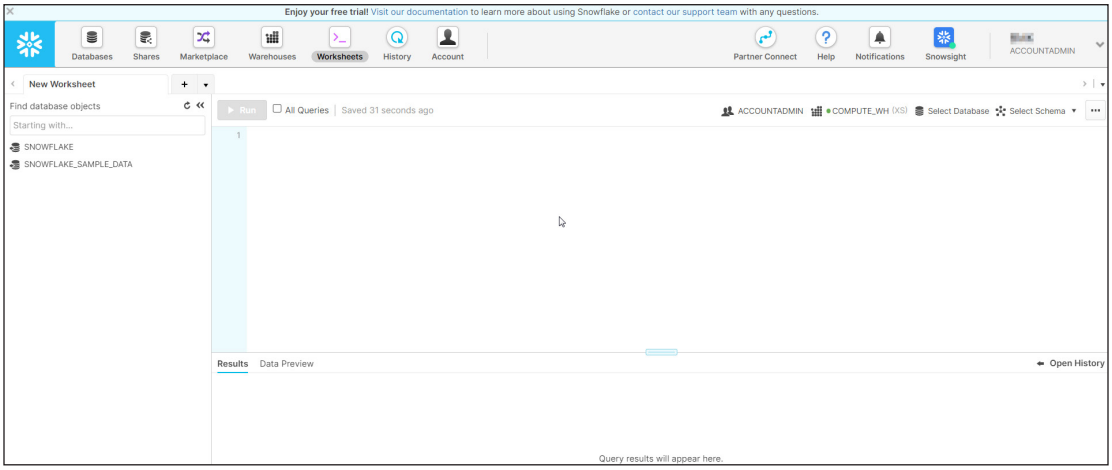

Figure 4.13: *Snowflake – Classic User Interface*

Based on the above screenshots, both Snowsight and Snowflake Classic Console mostly perform the same operation. However, Snowflake recommends using Snowsight as it has many evolving features that help the users easily navigate to different sections of the UI to query and analyze the data.

Snowflake connectors and drivers

Snowflake provides several methods to interact with Snowflake. The Partner Eco Systems consists of various third-party tools and technologies, and Snowflake-provided clients make this connection seamless and work in a few click integrations with Snowflake.

The Snowflake enables customers to access Snowflake through an extensive wide variety of industry-leading tools and technologies, drivers, connectors, utilities, and programming languages, including:

- **Snowflake client's tools**: SnowSQL (command line), Snowpark API, and so on.
- **Programmatic Interfaces/Drivers**: ODBC, JDBC, Python, GO, Node.js, Microsoft .NET, PHP, and so on.

How to get the latest snowflake connectors and drivers

There are two easy options available forgetting the latest Snowflake connectors and drivers. The options are:

- **Option 1: Snowsight**

In Snowsight, access the latest driver downloads by following these steps:

1. Click on your username located in the bottom left corner. This opens the account menu
2. Look for a section titled **Client Download**. This section is positioned near to My Profile section
3. Selecting **Client Download** will take you directly to the Snowflake Documentation (**https://docs.snowflake.com/en/user-guide/snowflake-client-repository**) page. There, you'll find links to the Snowflake Developer Center Download page (**https://docs.snowflake.com/en/user-guide/snowflake-client-repository#label-client-download-developer-center**) where you can download the most recent drivers.

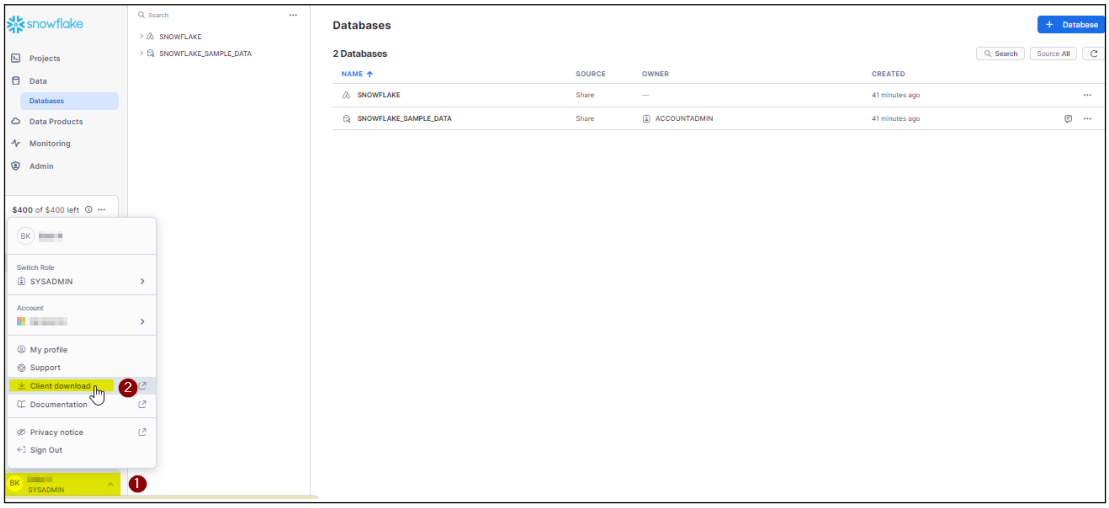

Figure 4.14: *Snowflake – Snowsight Client download*

- **Option 2: Snowflake Documentation Page**

To download the latest version of Snowflake drivers and connectors, users can directly navigate the Snowflake page: **https://docs.snowflake.com/en/user-guide/snowflake-client-repository**. There, you will find links to the Snowflake Developer Center Download page where you can download the most recent drivers: **https://docs.snowflake.com/en/user-guide/snowflake-client-repository#label-client-download-developer-center**.

The following figure references to the Snowflake Documentation Client download page:

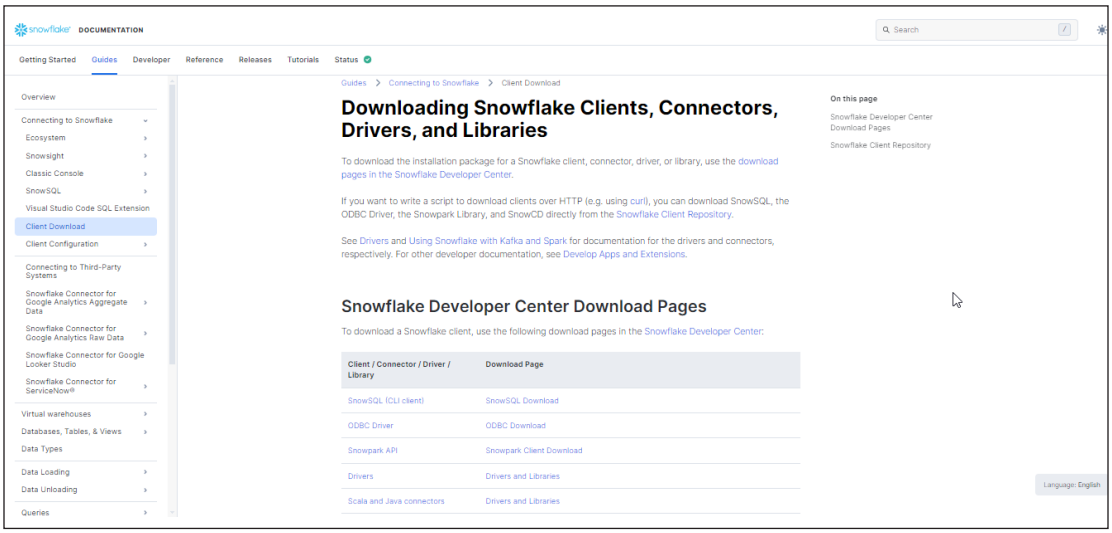

Figure 4.15: *Snowflake – Documentation/Developer Center Download Pages*

Snowpark

Snowpark is a revolutionary new feature within Snowflake that empowers data professionals to leverage familiar programming languages such as Python, Java, or Scala for advanced data manipulation and analysis directly within the Snowflake environment. This represents a significant leap beyond traditional SQL-based workflows, offering enhanced performance, flexibility, and scalability.

Snowpark offers several advantages over traditional SQL-only development, including:

- **Superior performance**: Snowpark harnesses the full power of Snowflake's optimized data processing engine, often achieving significantly faster execution times compared to pure SQL approaches.
- **Unparalleled flexibility**: Data professionals can utilize their preferred programming language and its associated libraries, unlocking a wealth of capabilities for advanced analytics and machine learning.

- **Simplified development**: Snowpark's intuitive API streamlines the development of complex data pipelines and transformations.
- **Seamless scalability**: Snowpark applications can effortlessly scale to handle massive datasets, ensuring optimal performance regardless of data volume.

How Snowpark works

Snowpark utilizes a DataFrame API, providing a familiar abstraction for data manipulation and transformation. DataFrames reside in memory, enabling rapid processing and analysis. Additionally, Snowpark supports custom **user-defined functions** (**UDFs**) and connectors to external data sources, further expanding its versatility.

If you are looking for a way to improve the performance, flexibility, and simplicity of your data processing workloads, then Snowpark is a great option to consider. Here are some specific examples of how Snowpark can be used:

- **Data scientists**: Build and deploy machine learning models directly within Snowflake, accelerating model development and deployment cycles.
- **Data engineers**: Develop sophisticated data pipelines and applications that harness the full power of Snowflake's real-time and batch processing capabilities.

Here are some additional details about Snowpark:

- Snowpark currently supports Java, Python, and Scala.
- Snowpark supports all of the data types that are supported by Snowflake.
- Snowpark libraries are available for the following platforms:
    - Windows
    - Linux
    - macOS

Snowpark represents a paradigm shift in data processing within Snowflake, offering a powerful and flexible toolkit for data professionals. By combining the performance of Snowflake with the versatility of popular programming languages, Snowpark empowers organizations to unlock new levels of data-driven insights and innovation.

Conclusion

In this chapter, we learned about how to sign up and get started with Snowflake in a few clicks, Snowflake's user interfaces, Snowsight, various drivers, and connectors available to connect Snowflake. We also learned about Snowflake's new feature Snowpark, and its benefits.

In the next chapter, we will learn about the Snowflake Catalogs and objects such as databases, schemes, tables, views, stored procedures, user-defined functions, and various data types available within Snowflake.

Points to remember

Refer to the following summary for a quick reference to what we learned so far in this chapter:

- **How long the Snowflake trial account is valid?**
    - The Snowflake trial account comes with a 30-day free trial that includes $400 worth of Snowflake usage to try, explore, and validate various use cases.
- **How to access the Snowflake trial account after signing up?**

  Snowflake offers a wealth of tools and interfaces to cater to different user preferences and needs. The following are the key user interfaces to perform various actions in Snowflake:
    - Snowsight: The Snowflake Web Interface
    - Classic Console (Available only to older accounts)
    - SnowSQL (CLI Client)
    - Snowflake Extension for Visual Studio Code
    - General Configuration (All Clients)
    - 3rd-party tools and technologies
    - Following are the Key Snowflake connectors and drivers:
        - **Snowflake Clients tools**: SnowSQL (command line), Snowpark API, etc.
        - **Programmatic Interfaces/Drivers**: ODBC, JDBC, Python, GO, Node.js, Microsoft .NET, PHP, and so on.
- **What are the two types of Snowflake UI and how do they differ from each other?**
    - **Snowsight- The Modern Look**: Think of it as a sleek, visually driven interface. Like a well-organized app, it uses tabs, menus, and buttons to let you easily manage your data.
    - **Classic Console - The Old Reliable**: It's the original interface, offering a more *text-based* approach. Think of it like a command prompt specifically for Snowflake. The Classic UI is available only to the older accounts.

  Both Snowsight and Snowflake Classic Console mostly perform the same operation. However, Snowflake recommends using Snowsight as it has many evolving features that help the users easily navigate to different sections of the UI to query and analyze the data.
- **What is Snowpark?**

  Imagine having a powerful tool that lets you analyze massive amounts of data directly within Snowflake, using your preferred programming language (Python, Java, or Scala). That's what Snowpark offers! Here's a quick rundown:

Snowpark is a set of libraries and tools built into Snowflake that allows you to:

  - **Run custom code**: Write functions, scripts, and even machine learning models directly within Snowflake, without moving your data around.
  - **Process data at scale**: Leverage Snowflake's powerful computing resources to handle massive datasets efficiently.
  - **Choose your language**: Work comfortably with Python, Java, or Scala, whichever you prefer.

- List a few key features and benefits of Snowpark.
  - **Language freedom**: Write complex logic using your preferred language. Snowpark currently supports Java, Python, and Scala.
  - **Seamless integration**: Run your code directly within Snowflake, no need for separate clusters or data movement.
  - **Automatic scaling**: Snowflake takes care of scaling your resources up or down as needed, so you can focus on your code.
  - **Security and governance**: Enjoy the same enterprise-grade security and governance that Snowflake offers for all your data.
- The following are the benefits:
  - **Faster development**: Write code quickly and iterate easily without worrying about data movement.
  - **Reduced costs**: No need for additional infrastructure or data transfer charges.
  - **Increased flexibility**: Solve complex problems with custom logic that goes beyond SQL.
  - **Improved collaboration**: Share and reuse code across teams and languages.

CHAPTER 5
Snowflake Catalogs and Objects

Introduction

In this chapter, we will learn about the Snowflake Catalogs and objects within the hierarchy, such as databases, schemas, tables, views, and various data types available within Snowflake. In addition, we will learn more details about various other objects in Snowflake like Stored Procedures, User Defined Functions, Snowpipe, Streams, Tasks, Shares, and Sequences to simplify and automate the data ingestion process.

Structure

In this chapter, we will discuss the following topics:

- Snowflake DB objects
- Snowflake data types
- Understanding Snowflake table structures
- Snowflake views
- Stored procedures
- User-Defined Functions
- SnowPipe
- Streams and tasks

- Shares
- Sequences
- Data storage monitoring

Objectives

By the end of this chapter, we aim to understand Snowflake catalogs and the components within its hierarchy, including databases, schemes, tables, views, and various available data types in Snowflake. In addition, readers learn to simplify and manage the data ingestion process using Snowflake's Stored Procedures, User Defined Functions, Snowpipe, Streams, Tasks, Shares, and Sequences.

Snowflake DB objects

Snowflake DB objects are the logical constructs that store data in Snowflake. They are organized in a hierarchical structure, with databases at the top, schemas below them, and object like tables and views below schemas. Following is the hierarchy of Snowflake object structure:

- **Databases** are the top-level containers for objects in Snowflake. Each database belongs to a single Snowflake account.
- **Schemas** are logical groupings of database objects, such as tables, views, and sequences. Each schema belongs to a single database.
- **Objects** are the smallest unit of storage in Snowflake. They can be tables, views, sequences, or other types of objects.

Snowflake DB objects are used to organize data and make it easier to manage. They also provide a way to control access to data by assigning different users and roles different permissions to objects.

Here is a list of the most common Snowflake DB objects:

- **Tables** are the most common type of object in Snowflake. They store data in a tabular format with rows and columns.
- **Views** are virtual tables that are based on the results of a SQL query. They do not store any data themselves, but they can be used to query data from other tables.
- **Sequences** are objects that generate a sequence of numbers. They are often used to generate unique IDs for rows in tables.

Other types of Snowflake DB objects include:

- Functions
- Stored procedures

- Virtual warehouses
- Data share
- File formats, named stages
- Snowflake extended objects - Tasks, streams, and so on

Snowflake DB objects provide a powerful and flexible way to store and manage data. They can be used to organize data, control access to data, and perform complex queries.

Snowflake data types

Snowflake supports a wide range of data types to use for column, expressions, parameters, local variables, and any other appropriate locations within Snowflake, including:

Numeric data types

Following are the supported numeric data types used to store number values:

- **`NUMBER`**: Default precision and scale are (38,0)
- **`INT, INTEGER, BIGINT, SMALLINT, TINYINT, BYTEINT`**: Synonymous with **`NUMBER`** except precision and scale cannot be specified. Refer the following details for various numeric data types and its range:
    - **`INT/INTEGER`**: Stores numbers from -2147483648 to 2147483647.
    - **`TINYINT`**: Stores numbers from -128 to 127.
    - **`SMALLINT`**: Stores numbers from -32768 to 32767.
    - **`BIGINT`**: Stores numbers from -9223372036854775808 to 9223372036854775807.
- **`DECIMAL`**: Stores decimal numbers with a precision of up to 38 digits and a scale of up to 18 digits. This decimal data type can alternatively represent as **`DECIMAL`** or **`DEC`** or **`NUMERIC`**
- **`FLOAT, FLOAT4, FLOAT8`**: Stores floating-point numbers.
- **`DOUBLE, DOUBLE PRECISION, REAL`**: Synonymous with **`FLOAT`**

String data types

Following are the supported String data types used to store text values:

- **`VARCHAR`**: Stores variable-length text. The maximum length of a **`VARCHAR`** column is 16,777,216 bytes.
- **`CHAR, CHARACTER`**: Stores fixed-length text. The maximum length of a **`CHAR`** column is 255 bytes.
- **`STRING, TEXT`**: Synonymous with **`VARCHAR`**

Binary data types

Following are the supported Binary data types used to store binary data:

- **`BINARY`**: Stores binary data.
- **`VARBINARY`**: Stores variable-length binary data. Synonymous with **`BINARY`**

Logical data types

Snowflake only has one logical data type: **`BOOLEAN`**. This means it can represent only two possible values:

- **`TRUE:`** This represents something that is true, exists, or is on.
- **`FALSE:`** This represents something that is false, does not exist, or is off.

Here are some examples of how **`BOOLEAN`** data types are used in Snowflake:

- Is it raining today? (Possible answers: **`TRUE`** or **`FALSE`**)
- Do you like chocolate? (Possible answers: **`TRUE`** or **`FALSE`**)

Note – `BOOLEAN` data type is only supported for accounts provisioned after January 25, 2016.

Date and time data types

Following are the supported date and time data types used to store date/time values:

- **`DATE`**: Stores dates.
- **`DATETIME`**: Alias for **`TIMESTAMP_NTZ`**
- **`TIME, TIMESTAMP`**: Stores dates and times. Alias for one of the **`TIMESTAMP`** variations (**`TIMESTAMP_NTZ`** by default)
- **`TIMESTAMP_TZ`**: **`TIMESTAMP`** with time zone
- **`TIMESTAMP_NTZ`**: **`TIMESTAMP`** with no time zone; time zone, if provided, is not stored.
- **`TIMESTAMP_LTZ`**: Stores dates and times with a local time zone; time zone, if provided, is not stored.

In addition to these basic data types, Snowflake also supports a number of other data types, such as semi-structured data types to store data like JSON and geospatial data types.

Semi-structured data types

Following are the supported Semi- structured data types used to store JSON, XML, and so on:

- **`VARIANT`**: Stores hierarchical data. A **`VARIANT`** column can contain a mixture of different data types, including other **`VARIANT`** columns. The maximum length of

a **VARIANT** is 16 MB. A **VARIANT** can store a value of any other type, including **OBJECT** and **ARRAY**.

- **OBJECT**: An **OBJECT** contains semi-structured like JSON key-value pairs. The maximum length of an **OBJECT** is 16 MB.
- **ARRAY**: Snowflake **ARRAY** is similar to **ARRAY** in other DB / Programming Languages.

Geospatial data types

Snowflake offers native support to geospatial structures such as lines, points, and polygons on the Earth's surface. Following are the supported geospatial data types used to store geospatial values:

- GEOGRAPHY
- GEOMETRY

VECTOR data types

Snowflake supports a single vector data type, **VECTOR**. This data type facilitates applications requiring semantic vector search and retrieval, including **Retrieval-Augmented Generation** (**RAG**). Additionally, it supports efficient vector operations, making it suitable for vector-processing applications. Currently this is only supported in SQL, Python connector and Snowpark Python Library. Not supported by other programming languages.

Understanding Snowflake table structures

A Snowflake table is a logical construct that stores data in a tabular format. It is composed of rows and columns, with each row representing a single record and each column representing a single attribute of that record.

The structure of a Snowflake table is defined by the following elements:

- **Name**: The name of the table.
- **Schema**: The schema in which the table resides.
- **Columns**: The columns in the table, which define the attributes of the records stored in the table.
- **Data types**: The data types of the columns in the table.
- **Constraints**: The constraints on the columns in the table, such as NOT NULL, UNIQUE, and CHECK constraints.

The following is an example of a Snowflake table structure:

```
SQL:
-- Sample Syntax/Code for creating Table:
create or replace table CUSTOMERS (
```

```
    id INT NOT NULL,
    name VARCHAR(255),
    email VARCHAR(255),
    PRIMARY KEY (id)
);
```

This table has four columns:

- **id**: An **INT** column that is the primary key of the table.
- **name**: A **VARCHAR(255)** column that stores the name of the record.
- **email:** A **VARCHAR(255)** column that stores the email address of the record.
- **PRIMARY KEY**: A constraint that specifies that the id column is the primary key of the table.

The structure of a Snowflake table can be modified using the **ALTER TABLE** statement. For example, the following statement would add a new column to the **CUSTOMERS** table:

```
SQL:
-- Sample Syntax/Code for adding new columns to the existing Table:
ALTER TABLE CUSTOMERS ADD COLUMN city VARCHAR(255), STATE VARCHAR(255), ZIP
NUMBER ;
```

The structure of a Snowflake table can also be viewed using the **SHOW COLUMNS** statement. For example, the following statement would show the columns in to the **CUSTOMERS** table:

```
SQL:
-- Sample Syntax/Code to View COLUMNS in a table:
SHOW COLUMNS IN CUSTOMERS;
-- Alternate method to View COLUMNS in a table:
SHOW COLUMNS IN TABLE CUSTOMERS;
```

Let us understand the different types of Snowflake tables. Snowflake tables are further classified into internal and external tables. Snowflake offers three types of internal tables: permanent, temporary, and transient. Here are some additional details about different Snowflake tables:

Permanent tables

Imagine a sturdy storage box. This is your classic Snowflake table, built to last. Once created, it exists indefinitely until you explicitly drop it. Following are few additional factors about permanent tables:

- Permanent tables are the most common type of table in Snowflake.
- Permanent tables are the default type of table in Snowflake.
- They are stored in the Snowflake database and can be accessed by all users who have access to the database.

- They are used to store data that needs to be persisted over time, meaning that they are not deleted when the session ends.
- Permanent tables can be used to store data for reporting, analysis, and other purposes.
- Time travel retention period 0 or 1 for Snowflake Standard Edition, 0 to 90 for Enterprise Edition and above. The *Fail-safe* retention period is 7 days.

Transient tables

Think of these as shared whiteboards. Transient tables are sticking around until drop them and are visible to anyone with the appropriate access privileges. Following are few additional factors about Transient tables:

- Transient tables offer storage option similar to permanent tables; however this does not provide *Fail-safe* for extended data protection. It is important to note that due to the lack of a fail-safe mechanism, once data is removed from a transient table, it cannot be retrieved.
- Transient tables are a good choice for temporary data manipulation, ensuring its availability throughout the designated timeframe but without the overhead of long-term data recovery mechanisms.
- Transient tables can have a Time Travel retention period of either 0 or 1 day

```
SQL:
-- Sample Syntax/Code for creating Transient Table:
CREATE TRANSIENT TABLE mysnowtranstable (id NUMBER, creation_date DATE);
```

Temporary tables

Think of a paper bag, handy for short-term use. Temporary tables live only within your current session and dropped automatically once you log out. Use them for quick calculations, intermediate results, or data you do not need to keep around. Following are few additional factors about temporary tables:

- Temporary tables are used to store data that is only needed for the current session.
- Only visible to the user who created them in the current session.
- Temporary tables are deleted when the session ends.
- Temporary tables are a good way to store intermediate results of a query or to store data that is not needed after the session ends.
- Temporary tables can also have a Time Travel retention period of 0 or 1 day. However, the retention period ends as soon as the table is dropped or the session in which the table was created ends.
- Temporary tables have no Fail-safe period

```
SQL:
-- Sample Syntax/Code for creating Temporary Table:
CREATE TEMPORARY TABLE mysnowtemptable (id NUMBER, creation_date DATE);
```

Here is a table that summarizes the different types of Snowflake tables with the key differences:

Table Type	Visibility	Persistence	Cloning (Table Source Type to Target Type)	Time Travel Retention Period (Days)	Fail-safe Period (Days)	Min, Max Historical Data Maintained (Days)
Permanent	All users	Until explicitly drooped	Permanent => Permanent Permanent => Transient Permanent => Temporary	- 0 or 1 For Std Edition - 0 to 90 For Enterprise Ed and higher	7	7, 8 7, 97
Transient	Users with appropriate access	Until explicitly drooped	Transient => Transient Transient => Temporary	0 or 1	NO	0, 1
Temporary	Current user	Reminder of the session	Temporary => Temporary Temporary => Transient	0 or 1	NO	0, 1

Table 5.1: *Snowflake – Comparison of different table types*

Snowflake external tables

Snowflake external tables provide a mechanism to seamlessly access data residing in external cloud storage locations like Amazon S3, Google Cloud Storage, or Azure. This functionality eliminates the need for data movement into Snowflake's native storage, potentially reducing storage costs.

Snowflake external table is a virtual table that references data files stored in an external stage. External tables facilitate the storage of essential file-level metadata within Snowflake. This metadata encompasses details such as filenames, version identifiers, and other relevant properties. External tables are read-only, meaning that you cannot insert, update, or delete data from them.

To create an external table, you need to specify the following information:

- The name of the external table.
- The schema in which the external table will be created.
- The name of the external stage that contains the data files.
- The path to the data files.
- The column definitions for the external table.

The following is an example of a Snowflake external table:

```
SQL:
-- Sample Syntax/Code for creating external Table:
CREATE OR REPLACE EXTERNAL TABLE mysnow_externaltable
(
  column1_name data_type,
  column2_name data_type,
  ...
)
WITH LOCATION = @my_external_stage
FILE_FORMAT = (FORMAT_NAME = my_file_format)
AUTO_REFRESH = TRUE; -- Optional: set to TRUE if you want Snowflake to
automatically refresh the metadata for the external table
```

This external table references data files that are stored in the **`my_external_stage`**. This could be one of the external cloud storage locations like Amazon S3, Google Cloud Storage, or Azure. Once you have created an external table, you can query it using the same syntax that you would use to query a regular Snowflake table. For example, the following query would select all of the rows from the **`my_external_table`** external table:

```
SQL:
-- Sample Syntax/Code to query external Table:
SELECT * FROM mysnow_externaltable;
```

External tables are a powerful way to access data that is stored in external storage. They can be used to query data from cloud storage services. By leveraging external tables, organizations can efficiently query data residing in external storage locations without the need for data movement. This streamlines data accessibility and promotes a centralized management approach.

Here are some of the benefits of using external tables in Snowflake:

- They allow you to access data that is stored in external storage.
- They are read-only, so you do not need to worry about accidentally updating or deleting data.
- They are easy to create and manage.

- They can be queried using the same syntax that you would use to query a regular Snowflake table.

If you need to access data that is stored in external storage, then external tables are a good option to consider. They are easy to create and manage, and they can be queried using the same syntax that you would use to query a regular Snowflake table.

Snowflake views

A Snowflake view is a virtual table that is based on the results of a SQL query. Views do not store any data themselves, but they can be used to query data from other tables.

Views are a powerful way to simplify complex queries and provide a layer of abstraction for data. They can also be used to restrict access to data, by creating views that only expose a subset of the data in a table.

To create a Snowflake view, you need to specify the following information:

- The name of the view.
- The SQL query that the view is based on.
- The schema in which the view will be created.

The following is an example of a Snowflake view:

```
SQL:
-- Sample Syntax/Code for creating View:
create or replace view my_view as
SELECT * FROM CUSTOMERS where STATE in ('TX')
;
```

This above query creates a view called **`my_view`** that is based on the results of the **`CUSTOMERS`** table. Once you have created a Snowflake view, you can query it using the same syntax that you would use to query a regular Snowflake table. For example, the following query would select all of the rows from the **`my_view`** view:

```
SQL:
-- Sample Syntax/Code to query the View:
SELECT * FROM my_view;
```

Here are some of the benefits of using views in Snowflake:

- They simplify complex queries.
- They provide a layer of abstraction for data.
- They can be used to restrict access to data.

If you need to simplify complex queries or provide a layer of abstraction for data, then views are a good option to consider. They are also a good way to restrict access to data.

Here are some additional details about Snowflake views:

- Views can be created in any schema.
- Views can be based on the results of any SQL query.
- Views can be updated or deleted at any time.
- Views can be used in any other Snowflake object, such as a stored procedure or a function.

Types of Snowflake Views

Snowflake supports the following various types of views to meet different users' needs.

- Non materialized (simply referred as views)
- Materialized views.
- Secure views

Here are some additional details about the different types of Snowflake views:

- **Non materialized views:**
    - Non materialized views are the most common type of view in Snowflake. They are based on the results of a SQL query, but they do not store any data themselves.
    - They are a good way to simplify complex queries and provide a layer of abstraction for data. They can also be used to restrict access to data, by creating views that only expose a subset of the data in a table.
- **Materialized views(MV):**
    - Materialized views required Snowflake Enterprise edition and above.
    - Materialized views are a good choice for storing data that needs to be accessed frequently and query result will not change often.
    - Materialized views behave more like a table and store a copy of the data that they are based on and updated automatically whenever the underlying data is updated.
    - Materialized views are automatically maintained by Snowflake. A background service updates the MV after the changes happened to the base table.
    - Materialized view can query only single table. Joins, including self joins are not supported by MVs.
    - Many aggregate and user defined functions are not allowed or supported my MVs.
- **Secure views:**
    - These views are a type of regular view that is restricted to certain users or

roles.

- Secure views can be used to control access to data.
- They are a good way to restrict access to sensitive data, such as financial data or customer data.
- The secure view definition is not visible all the users and only exposed to authorized users.
- The secure view can be a materialized or non-materialized view.

Stored procedures

A Snowflake stored procedure is a reusable block of code that can be called from SQL. Stored procedures are a powerful way to encapsulate business logic and improve the performance of queries.

Stored procedures can be written in any of the supported languages in Snowflake, including SQL, Python, Java, JavaScript and Scala. Stored procedures can be created in any schema.

To create a stored procedure, you need to specify the following information:

- The name of the stored procedure.
- The schema in which the stored procedure will be created.
- The language in which the stored procedure is written.
- The body of the stored procedure.

The body of the stored procedure can contain any valid SQL statement with in the code. For example complex calculations can be achieved easily by using the stored procedures.

Here is a sample stored procedure code in Snowflake:

```
SQL:
-- Sample Syntax/Code to create Stored Procedure:
CREATE OR REPLACE PROCEDURE return_greater_number(number_1 INTEGER,
number_2 INTEGER)
RETURNS INTEGER NOT NULL
LANGUAGE SQL
AS
$$
BEGIN
  IF (number_1 > number_2) THEN
    RETURN number_1;
  ELSE
    RETURN number_2;
  END IF;
```

```
END;
$$
;
```

This stored procedure takes two parameters: The first and second number as an input and then return the value of the greatest number. To call this stored procedure, you can use the following SQL statement:

```
SQL:
-- Sample Syntax/Code to execute/calling the Stored Procedure:
CALL return_greater_number(100, 500);
```

Here are some additional details about the sample stored procedure code:

- The **`CREATE OR REPLACE PROCEDURE`** statement creates the stored procedure if it does not already exist, or it replaces the stored procedure if it does exist.
- The **`LANGUAGE SQL`** clause specifies that the stored procedure is written in SQL.
- The **`BEGIN`** and **`END`** keywords mark the beginning and end of the body of the stored procedure.
- The **`RETURN`** statement returns the value of the stored procedure.

Here are some of the benefits of using stored procedures in Snowflake:

- They encapsulate business logic.
- They can improve the performance of queries.
- They can be reused.

If you need to encapsulate business logic or to improve the performance of queries, then stored procedures are a good option to consider. They are also a good way to reuse code.

Here are some additional details about Snowflake stored procedures:

- Stored procedures can be created in any schema.
- Stored procedures can be written in any of the supported languages in Snowflake.
- Stored procedures can be called from SQL.
- Stored procedures can call other stored procedures.

User-Defined Functions

A **User-Defined Function** (**UDF**) is a reusable block of code that can be called from SQL. UDFs are a powerful way to encapsulate business logic and improve the performance of queries.

UDFs can be written in any of the supported languages in Snowflake, including SQL, Python, Java, JavaScript and Scala. UDFs can be created in any schema.

To create a UDF, you need to specify the following information:

- The name of the UDF.
- The schema in which the UDF will be created.
- The language in which the UDF is written.
- The body of the UDF.

The body of the UDF can contain any valid SQL statement. UDFs can also call other UDFs.

Once you have created a UDF, you can call it from SQL. Here are some of the benefits of using UDFs in Snowflake:

- They encapsulate business logic.
- They can improve the performance of queries.
- They can be reused.

If you need to encapsulate business logic or improve the performance of queries, then UDFs are a good option to consider. They are also a good way to reuse code.

Here are some additional details about Snowflake UDFs:

- UDFs can be created in any schema.
- UDFs can be written in any of the supported languages in Snowflake.
- UDFs can be called from SQL.
- UDFs can call other UDFs.

Here is a sample UDF code in Snowflake:

```
SQL:
-- Sample Syntax/Code to create SQL UDF:
CREATE OR REPLACE FUNCTION convert_uppercase (input VARCHAR)
RETURNS VARCHAR
AS
$$
  SELECT UPPER(input)
$$
;
```

This UDF takes a name as input and returns the name in uppercase. The UDF returns a **VARCHAR** value. To call this UDF, you can use the following SQL statement:

```
SQL:
-- Sample Syntax/Code to call the UDF:
SELECT convert_uppercase ('Joe');
```

This statement would return the value **JOE**.

Here are some additional details about the sample UDF code:

- The **`CREATE`** or **`REPLACE`** function statement creates the UDF if it does not already exist, or it replaces the UDF if it does exist.
- The **`BEGIN`** and **`END`** keywords mark the beginning and end of the body of the UDF.
- The **`UPPER`** function returns the uppercase version of the input string.

SnowPipe

Snowpipe is Snowflake's fully-managed, serverless data ingestion service, designed to automate and streamline the loading of data into your Snowflake data warehouse. It functions as a continuous data pipeline, automatically detecting and ingesting files as soon as they are available in a designated cloud storage location (stage).

In Snowflake's ecosystem, a pipe is a named object designed to facilitate automated data ingestion through the Snowpipe service. It encapsulates a **`COPY`** statement, defining the source data location (stage) and the target table for seamless data transfer. The pipe enables continuous data loading as files become available in the stage, eliminating the need for manual intervention. Snowpipe supports a wide array of file types, including structured and semi-structured formats like JSON and Avro, ensuring compatibility with diverse data sources and formats.

Automating Snowpipe using cloud messaging

Snowpipe Automating continuous data loading using cloud messaging is a feature that allows you to automatically load data from files as soon as they're available in cloud storage. This is done by using cloud messaging services to notify Snowpipe when new files are added, triggering the loading process. Here's how it works:

1. **Event notification**: When a new file is added to your cloud storage (for example, Amazon S3, Google Cloud Storage, or Azure Blob Storage), an event notification is sent to a message queue.
2. **Snowpipe notification**: Snowpipe is configured to listen to this message queue. When it receives the notification, it knows that a new file is ready to be loaded.
3. **Automatic loading**: Snowpipe automatically starts the loading process, copying the data from the file into your Snowflake table.

This automation process happens continuously, and eliminates the need for manual intervention or scheduled tasks, ensuring that your data is always up-to-date and accessible for immediate analysis. It is a serverless solution, meaning you do not have to worry about managing any infrastructure. Additionally, the serverless architecture of Snowpipe ensures that you only pay for the resources consumed during data loading, making it a cost-effective solution for businesses of all sizes.

The following table illustrates the compatibility of Snowpipe automatic data loading feature, utilizing cloud storage event notifications, contingent upon the cloud platform hosting your Snowflake account:

Snowflake Account Host	Amazon S3	Google Cloud Storage	Microsoft Azure Blob storage	Microsoft Data Lake Storage Gen2	Microsoft Azure General-purpose v2
Amazon Web Services	✓	✓	✓	✓	✓
Microsoft Azure	✓	✓	✓	✓	✓
Google Cloud Platform	✓	✓	✓	✓	✓

Table 5.2: *Snowpipe - Supported Cloud Storage Services*

Advantages of Snowpipe

Here are some of the key benefits and advantages of using Snowpipe:

- **Serverless**: Snowpipe is a serverless service, so you do not have to manage any infrastructure.
- **Automation and continuous ingestion**: Eliminates the need for manual data loading or scheduled tasks, ensuring your data warehouse is always up-to-date.
- **Near real-time data availability**: Loads data in micro-batches as it arrives, providing near-real-time access to your information for timely analysis and decision-making.
- **Scalability and efficiency**: Serverless architecture eliminates the need for infrastructure management and scales seamlessly to accommodate varying data volumes. Costs are incurred only for the actual data loaded and storage used.
- **Broad compatibility**: Supports a wide range of data formats, simplifying integration with existing data pipelines and systems
- **Easy to use and configure**: Snowpipe is easy to use and configure. You can create a pipe in a few minutes and start loading data.
- **Cost-effective**: Snowpipe is a cost-effective way to load data into Snowflake. You only pay for the data that is loaded.

Limitations of SnowPipe

Overall, Snowpipe is a powerful tool for loading data into Snowflake. It is easy to use and can be configured to load data from a variety of Snowflake supported file formats. However, it is important to be aware of the limitations of Snowpipe before using it.

Here are some of the limitations of using Snowpipe:

- **Large files**: Snowpipe is not suitable for all data loads. While Snowpipe excels at handling smaller files and frequent updates, loading large files might require additional considerations for optimal performance.

- **Transformations**: Primarily designed for data ingestion, Snowpipe offers limited built-in capabilities for complex data transformations. Additional tools or processes may be required for extensive transformations.
- **Cost management**: While efficient, high-volume data loading or frequent batches can result in increased costs, particularly with large batch files load.

Snowpipe vs. bulk data loading

Snowpipe and bulk data loading are two distinct methods for ingesting data into Snowflake, each catering to different use cases and operational requirements. Choose Snowpipe for automated, continuous loading of data with a focus on near-real-time accessibility. Opt for bulk data loading when dealing with large datasets, infrequent updates, or when immediate access to the data is not a primary concern. The optimal choice depends on the specific requirements of your data ingestion pipeline and the desired trade-offs between automation, speed, and data volume.

Here are some of the key comparisons between Snowpipe and Bulk Data load:

Snowpipe:

- **Ideal for**: Continuous, near-real-time data ingestion scenarios.
- **Mechanism**: Leverages a serverless architecture to automatically load data in micro-batches from cloud storage as soon as it becomes available.
- **Strengths**: Automated and highly efficient for frequent data updates, minimizing latency and enabling timely access to insights.
- **Limitations**: May not be the most optimal choice for loading massive, one-time datasets due to micro-batch processing.
- **File size**: For best load performance loading very large files are not recommended, so split the large data files into smaller in size around 100-250 MB of compressed data. For Semi structured (`VARIENT`) can have a maximum size up to 16 MB of uncompressed for better performance. If data exceeds 16 MB use the STRIP_OUTER_ARRAY file format option for the COPY INTO command to remove the outer array structure that can load the records in separate rows in the table.
- **Load history:** Stored in the metadata of the pipe load for 14 days. User can find the information using `ACCOUNT_USAGE` view.
- **Compute resource**: Uses Snowflake supplied compute warehouse
- **Compute cost**: Billed based on the compute resource used in Snowpipe to load the data/files.

Bulk data loading:

- **Ideal for**: Large, infrequent data loads or situations where immediate data availability is not critical.

- **Mechanism**: Manually initiated process where data is loaded in a single batch operation.
- **Strengths**: Efficient for handling large volumes of data in a single transaction, potentially reducing overall processing time.
- **Limitations**: Requires manual intervention, less responsive to real-time data updates, and may introduce latency compared to Snowpipe.
- **File size**: For best load performance loading very large files are not recommended, so split the large data files into smaller in size around 100-250 MB of compressed data. For Semi structured (`VARIENT`) can have a maximum size up to 16 MB of uncompressed for better performance.
- **Load history**: Stored in the metadata of the target table for 64 days. Available upon completion of the `COPY` statement output
- **Compute resource**: Required user supplied compute warehouse
- **Compute cost**: Billed based on the amount of the time each virtual Warehouse is active

Streams and tasks

Streams and tasks can be used together to create a continuous **Extract, Load, Transform** (**ELT**) pipeline. The stream can be used to capture changes to data in a table, and the task can be used to run a SQL statement or stored procedure on the stream data. This allows you to automate the ELT process and ensure that your data is always up to date.

Here is some information about Snowflake streams and tasks:

- Snowflake Streams are a powerful feature that can be used to capture changes to data in real time. They are created on top of existing tables and track all inserts, updates, and deletes that occur. When a change is made to a table, Snowflake creates a new row in the stream. The stream row contains information about the change, such as the old and new values, the timestamp of the change, and the user who made the change.
- Tasks are a Snowflake feature that allows you to run a SQL statement or stored procedure on a schedule. Tasks can be used to perform a variety of tasks, such as loading data, running reports, or sending emails.

Here are some of the benefits of using streams and tasks together:

- **Automated ELT:** Streams and tasks can be used to automate the ELT process. So, no manual effort to manage the data loads.
- **Consistent data:** Streams and tasks can be used to ensure that your data is always up to date. This is important for ensuring the accuracy of your reports and analyses.
- **Scalable:** Streams and tasks can be scaled to meet the needs of your business. This means that you can use them to process large amounts of data without having to worry about performance.

Overall, Snowflake Streams and Tasks are powerful features that can help you to build efficient, scalable, and reliable data pipelines. If you need to automate the ELT process or ensure that your data is always up to date, then streams and tasks are a good option to consider. They are easy to use and can be scaled to meet the needs of your business. Here are some additional details about Snowflake streams and tasks:

Additional details about streams:

- Streams are created on tables.
- Streams are read-only.
- Streams are automatically created when a change is made to a table.
- Streams can be queried using the **`SELECT`** statement.

Additional details about tasks:

- Tasks are created on schedules.
- Tasks can run SQL statements or stored procedures.
- Tasks can be scheduled to run at specific times or intervals.
- Tasks can be monitored to track their status.

Shares

Snowflake Data Share facilitates secure and efficient data collaboration within and outside your organization. Leveraging Snowflake's cloud-native architecture, Data Share enables seamless sharing of selected database objects without the need for data copying or transfer.

To establish a data share, you define its parameters, including:

- **Share name**: A unique identifier for reference.
- **Schema**: The designated location within Snowflake to house the share.
- **Objects**: The specific tables, views, or other data elements to be included.
- **Permissions**: Granular access controls specifying what actions each recipient can perform.

Once configured, authorized users are granted access via the share name and associated permissions, enabling them to seamlessly integrate shared data into their workflows.

Here are some of the benefits of using Snowflake data share:

- **Optimized resource utilization**: Eliminating redundant data copies reduces storage costs and enhances performance.
- **Enhanced security**: Granular permissions ensure that data access is restricted to authorized individuals and actions.
- **Simplified management**: Intuitive controls streamline the creation, modification, and revocation of data shares.

Snowflake data share is a robust solution for organizations seeking to unlock the full potential of their data assets through secure, controlled collaboration.

Sequences

Snowflake sequences offer a robust and efficient mechanism for generating unique, sequential numeric identifiers within a database environment. Primarily employed to establish primary keys for tables, their versatility extends to other use cases requiring distinct identifiers such as order numbers or customer IDs.

Sequence creation is facilitated by the **`CREATE SEQUENCE`** command, while the **`NEXTVAL`** function effortlessly retrieves the subsequent number in the sequence.

Here are some of the benefits of using Snowflake sequences:

- **Streamlined implementation:** Sequences are simple to create, manage, and integrate into existing workflows.
- **Enhanced performance:** The generation of unique numbers is optimized for efficiency, contributing to improved resource utilization.

Here are some of the use cases for Snowflake sequences:

- **Primary key generation:** Ensures the uniqueness of each record within a table, a fundamental principle of relational database design.
- **Unique identifier creation:** Facilitates the assignment of distinctive identifiers to entities like orders, invoices, or customer accounts.

In summary, Snowflake sequences provide a valuable tool for database administrators and developers seeking a reliable and high-performance solution for generating unique, sequential numbers to support diverse data management requirements.

Data storage monitoring

Regular monitoring of Snowflake data storage and compute is a recommended best practice to ensure optimal cost control and prevent unanticipated expenses for the organization. As the Snowflake account administrator (**`ACCOUNTADMIN`** role), user have access to comprehensive data storage insights through both Snowsight and the Classic Console. Following are the navigation steps to view Snowflake Cost Management.

- **Snowsight:** Navigate to **Admin** | **Cost Management** | **Consumption** for a detailed overview of your account's total average data storage usage, along with granular breakdowns for databases, internal and named stages, and Fail-safe data.
- **Classic Console** (Note: Available only to the older accounts)**:** Access the same comprehensive storage information via the **Account** tab | **Billing & Usage** | **Average Storage Used**.

By leveraging Snowflake Cost Management feature, you can effectively monitor and manage your organization's data storage consumption within the Snowflake environment. In addition, Snowflake also provides a number of metrics that you can use to track data storage usage, including:

- **`DATABASE_STORAGE_USAGE_HISTORY`**: This view returns the average daily storage usage for a single database or all databases in your account within a specified date range. The results include all data stored in tables and materialized views and historical data maintained in fail-safe for the database(s).
- **`STAGE_STORAGE_USAGE_HISTORY`**: This view returns the average daily storage usage for a single stage or all stages in your account within a specified date range. The output will include the storage result for named internal stages and default staging area (for tables and users). This function returns stage storage usage within the last 6 months.
- **`TABLE_STORAGE_METRICS`**: This view provides a detailed breakdown of storage utilization at the table level. This information is used to calculate the storage billing for each table in the account, including tables that have been dropped but continue to incur storage costs. In addition to table metadata, this view also provides a number of metrics about data storage for individual tables, including the total storage used by active and deleted bytes(Time Travel, Fail-safe and retained for clones) that are still accruing storage cost.

Here are some of the metrics that you can track to monitor data storage in Snowflake. By tracking these metrics, you can get a better understanding of how your data storage is being used and identify any potential problems.

- **Storage used:** This metric tracks the amount of storage that is currently being used by your data warehouse.
- **Storage capacity:** This metric tracks the total amount of storage that is available in your data warehouse.
- **Storage growth rate:** This metric tracks the rate at which storage is being used in your data warehouse.
- **Storage trends:** This metric tracks the historical trends in storage usage in your data warehouse.

By monitoring your data storage usage, you can identify areas where you can reduce your storage costs. For example, you can identify tables that are no longer needed and delete or drop permanently to reduce the storage cost.

Here are some of the best practices for monitoring data storage in Snowflake. By following these best practices, you can ensure that your data storage is being monitored effectively:

- **Set up alerts:** You can set up alerts to notify you when your storage usage reaches a certain threshold. This will help you to identify potential problems before they become too costly.

- **Use the right metrics:** Not all metrics are created equal. When monitoring your data storage usage, you should focus on the metrics that are most relevant to your business needs.
- **Track your trends:** It is important to track your data storage usage over time so that you can identify trends. This will help you to make informed decisions about how to manage your storage costs. This can help you to identify any potential problems early on.
- **Snowflake Audit Logs:** Snowflake also provides audit logs that can be used to track changes to your data warehouse. The audit logs can be used to track data storage changes, such as when data is added, updated, or deleted.
- **Data Retention Setup:** In certain scenarios where the storage costs associated with high-churn dimension tables become too expensive, an alternative approach to **Change Data Capture** (**CDP**) may be preferred. For large dimension tables experiencing frequent changes and incurring excessive CDP costs, a viable solution is to create these tables as transient, with no Time Travel retention (`DATA_RETENTION_TIME_IN_DAYS=0`), and periodically copy their contents into a permanent table.
- **Third-Party Tools:** There are a number of third-party tools that can be used to monitor data storage in Snowflake. These tools can provide additional metrics and insights that are not available in the Snowflake monitoring tool.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **How DB objects are organized in Snowflake?**

  Snowflake DB Objects are the logical constructs that store data in Snowflake. They are organized in a hierarchical structure, with databases at the top, schemas below them, and object like tables and views below schemas.

- **List the most common Snowflake DB Objects in Snowflake:**
  - Databases, schemas, and tables are the most common type of object in Snowflake.
  - Views are virtual tables that are based on the results of a SQL query
  - Sequences are objects that generate a sequence of numbers.
  - Other types of Snowflake DB Objects include, Stored procedures, User Defined Functions (UDFs), and Sequences, etc.
- **List various data types supported by Snowflake:**
  - **Numeric data types:**
    - `NUMBER`, `INT`, `INTEGER`, `BIGINT`, `SMALLINT`, `TINYINT`, `BYTEINT`, `DECIMAL`, `FLOAT`, `FLOAT4`, `FLOAT8`, `DOUBLE`, `DOUBLE PRECISION`, `REAL`

  - **String data types:**
    - **`VARCHAR`**, **`CHAR`**, **`CHARACTER`**, **`STRING`**, **`TEXT`**
  - **Binary data types:**
    - **`BINARY`**, **`VARBINARYBLOB`**
  - **Logical data types:**
    - **`BOOLEAN`**
  - **Date and time data types:**
    - **`DATE`**, **`DATETIME`**, **`TIME`**, **`TIMESTAMP`**, **`TIMESTAMP_TZ`**, **`TIMESTAMP_NTZ`**, **`TIMESTAMP_LTZ`**
  - **Semi-structured data types:**
    - **`VARIANT`**, **`OBJECT`**, **`ARRAY`**
    - The maximum length of a **`VARIANT`** and **`OBJECT`** is 16 MB.
  - **Geospatial data types data types:**
    - **`GEOGRAPHY`**, **`GEOMETRY`**

- **What are the different types Snowflake tables?**
  - Snowflake offers three types of physical tables: permanent, temporary, and transient.
- **What is snowflake external table and when this will be used?**
  - A Snowflake external table is a virtual table that references data files stored in an external stage. External tables are read-only, meaning that you cannot insert, update, or delete data from them.
  - External tables are a good way to access data that is stored in a cloud storage service, such as Amazon S3, Azure Blob or Google Cloud Storage.
- **List the types of Snowflake views and use case of the views.**
  - **Non materialized or regular views:**
    - Regular views are the most common type of view in Snowflake. They are based on the results of a SQL query, but they do not store any data themselves.
  - **Materialized views:**
    - Materialized views are a good choice for storing data that needs to be accessed frequently. They can also be used to improve the performance of queries that access the underlying data.
  - **Secure views:**
    - These views are a type of regular view that is restricted to certain users or roles.
    - Secure views can be used to control access to data.

- **Describe Snowflake Stored Procedure**
    - A Snowflake stored procedure is a reusable block of code that can be called from SQL.
    - Stored procedures can be written in any of the supported languages in Snowflake, including SQL, Python, Java, and Scala. Stored procedures can be created in any schema.
- **Describe User-defined functions**
    - A **User-defined function** (**UDF**) is a reusable block of code that can be called from SQL.
    - UDFs can be written in any of the supported languages in Snowflake, including SQL, Python, Java, and Scala. UDFs can be created in any schema.
- **What is Snowpipe?**
    - Snowpipe is a serverless data ingestion service offered by Snowflake.
    - Snowpipe automatically loads data from files as soon as they are available in a stage. This means you can load data from files in micro-batches, making it available to users within minutes, rather than manually executing COPY statements on a schedule to load larger batches.
- **Describe Snowflake streams and tasks:**
    - Streams are a Snowflake feature that allows you to track changes to data in a table. When a change is made to a table, Snowflake creates a new row in the stream. The stream row contains information about the change, such as the old and new values, the timestamp of the change, and the user who made the change.
    - Tasks are a Snowflake feature that allows you to run a SQL statement or stored procedure on a schedule. Tasks can be used to perform a variety of tasks, such as loading data, running reports, or sending emails.

Conclusion

In this chapter, we learned about the Snowflake Catalogs and objects within the hierarchy, such as databases, schemas, tables, views, and various data types available in Snowflake. We also learned about various other objects within Snowflake, like Stored Procedures, User Defined Functions, Snowpipe, Streams, Tasks, Shares, and Sequences to simplify the data ingestion process and how the data is organized in Snowflake. This chapter will help the readers summarize overall learning and keynotes to remember for the exam.

In the next chapter, we will learn about the Snowflake Security and security around the topics like security principles, how to set up the Snowflake accounts access, and various network security policies, multi-factor authentication, and federated authentication of the Snowflake.

Chapter 6
Snowflake Account Access

Introduction

In this chapter, we will learn about the Snowflake Security principles, how to set up the Snowflake account access, and various network security policies to secure the account from unauthorized access. In addition, we will learn the details about the **single sign-on (SSO**), **Multi-factor Authentication** (**MFA**), and Federated authentication of Snowflake.

Structure

In this chapter, we will cove.r the following topics:

- Outline security principles
- Account access/setup
- Network security and policies
- Single sign-on
- MFulti-Factor Authentication
- Federated authentication

Objectives

By the end of this chapter, the readers will gain knowledge of the snowflake security principle, account access setup, and how to implement diverse network security policies to safeguard the account against unauthorized access.

Outline security principles

Snowflake security principles are designed to protect your data from unauthorized access, modification, or deletion. These principles are implemented using a variety of security controls, including:

- **Data encryption**: Snowflake uses the industry-standard AES 256-bit encryption for robust protection. The data automatically encrypts at rest, meaning it's scrambled when stored; even the cloud provider cannot access the data. The data gets encrypted while moving between you and Snowflake (in transit) using TLS, a secure communication protocol. This means that your data is protected even if it is stored on Snowflake's servers or transmitted over the internet.
- **User authentication and authorization**: Users must be authenticated before they can access Snowflake. Authentication can be done using passwords, federated authentication, or OAuth. Once users are authenticated, they are granted access to specific resources based on their roles.
- **Network security**: Snowflake provides several features to help secure your network, including network policies, private connectivity, and firewall rules. Network policies allow you to control which IP addresses can access your Snowflake account. Private connectivity allows you to connect to Snowflake from your on-premises network. Firewall rules allow you to control which ports are accessible from the internet.
- **Access control**: Snowflake provides a granular access control approach that combines **Role-Based Access Control (RBAC)** and **Discretionary Access Control (DAC)** for robust data security. RBAC efficiently manages permissions by assigning them to pre-defined roles (e.g., analyst, scientist), which are then granted to individual users. DAC empowers data owners with the ability to grant specific access privileges to these roles, ensuring granular control over sensitive information. This hybrid approach provides a balance between centralized administration and the adaptability required for complex data governance.
- **Auditing**: Snowflake provides a comprehensive auditing system that allows you to track all activity in your account. This includes who accessed what data, when they accessed it, and what they did with it. The auditing system can be used to detect unauthorized access or malicious activity.

- **Compliance:** Snowflake is committed to compliance with a wide range of industry standards, including HIPAA, PCI DSS, and SOC 2. This means that Snowflake has implemented several security controls to meet the requirements of these standards.

In addition to these principles, Snowflake also provides many other security features, such as dynamic data masking, and role-based access control.

Account access setup

In this section, we will learn about the various steps performed to set up Snowflake account access. Refer to the following details for further steps involved in this process.

How to set up Snowflake account access

Snowflake is a cloud-based data warehouse that provides secure and scalable access to data. To set up Snowflake account access, you will need to create an account, create users, grant roles to users, and configure access control.

Creating an account

To create a Snowflake account, you will need to provide a unique account name, a login name, and a password. You can also choose to specify an email address and a default role for the account.

Creating users

Users are the individual accounts that will be accessing Snowflake. When you create a user, you will need to specify a username, a login name, and a password. You can also choose to specify an email address for the user.

Granting roles to users

Roles define what permissions a user has in Snowflake. When you grant a role to a user, you are giving the user the permissions that are associated with the role. There are several different roles that you can grant to users, such as the **`ACCOUNTADMIN`** role, the **`SYSADMIN`** role, and the **`DATAOWNER`** roles. There are predefined roles that can be used to grant to the user. You can also create custom roles as per your requirements and grant custom roles to the users.

Configuring access control

Access control defines who has access to what in Snowflake. You can configure access control for securable objects in your account, such as databases, schemas, and tables. When you configure access control, you are specifying which users have access to which objects and what permissions they have on those objects.

Tips for setting up Snowflake account access

Following are the tips to set up Snowflake account access:

- **Designate at least two account administrators**. This will help your account always be backed up with at least one person who can perform account-level tasks, even if one of the account administrators is unable to log in.
- **Use strong passwords and enable multi-factor authentication**. This will help protect your Snowflake account from unauthorized access.
- **Grant the minimum permissions that the user needs to do their jobs**. This will reduce the risk of unauthorized access and prevent access to sensitive data.
- **Review your access control settings regularly**. This will help ensure that your account is secure and that only authorized users have access to data.

Snowflake access control framework

Snowflake leverages a robust access control framework that blends two well-established models: **Role-Based Access Control** (**RBAC**) and **Discretionary Access Control** (**DAC**). This hybrid approach offers a granular and secure way to manage user permissions within the Snowflake environment. By combining RBAC and DAC, Snowflake achieves a balance between security and flexibility.

Following are the high-level summary of how the RBAC and DAC are used to secure the objects in Snowflake.

- RBAC establishes a foundation for efficient permission management. Permissions are assigned to predefined roles that align with business functions (for example, data analyst, data scientist). These roles are then granted to individual users, streamlining administration, and ensuring adherence to the principle of least privilege.
- DAC empowers data owners with granular control over their data assets. They can grant specific access privileges to the RBAC roles, enabling them to tailor permissions based on user needs and data sensitivity. This fosters data ownership and accountability within the organization.

In Snowflake, access to securable objects is allowed via privileges assigned to roles, which are, in turn, assigned to users or other roles. Granting a role to another role creates a role hierarchy, which allows privileges to be inherited by child roles. In addition, each securable object has an owner that can grant access to other roles. The key concepts of Snowflake's access control framework are:

- **Securable objects**: These are the objects that can be secured in Snowflake. Examples of securable objects include databases, schemas, tables, views, and functions.

- **Roles**: These are the entities to which privileges can be granted. Roles are assigned to users, who can then use the privileges granted to the role to access securable objects.
- **Privileges**: These define the level of access that a user has to a securable object. There are a variety of privileges that can be granted to a role, such as the ability to create, read, update, and delete data.
- **Users**: These are the entities that are granted roles. Users can be individuals or groups of users.

The Snowflake Access Control Framework provides a high degree of flexibility and control over access to securable objects. By using roles and privileges, you can precisely define who has access to what and what they can do with it.

Benefits of Snowflake's access control framework

The Snowflake access control framework offers several benefits, including:

- **Granular control**: You can precisely define who has access to what and what they can do with it.
- **Flexibility**: You can easily add or remove users and roles, and you can change privileges as needed.
- **Security**: The Snowflake Access Control Framework is designed to be secure and to protect your data.

The Snowflake Access Control Framework is a powerful and flexible way to manage access to your data in Snowflake. If you are looking for a secure and efficient way to control who has access to your data, then the Access Control Framework is a great option.

For more information refer to the *Additional references* sections for best practices on how to set up the Snowflake access control framework.

Network security and policies

Snowflake provides a comprehensive set of network security features and policies to help protect your data. These features and policies are designed to prevent unauthorized access to your Snowflake account and data, both while it is in transit and at rest.

The creation of network policies is restricted to security administrators or higher roles with the designated "CREATE NETWORK POLICY" privilege. These policies can be implemented at the account, security integration, or user level. However, it's important to note that existing network policies that are currently assigned cannot be replaced using the "CREATE OR REPLACE NETWORK POLICY" command.

Network security features

The following are the network security features:

- **Network policies**: Network policies allow you to restrict access to your Snowflake account based on the IP address of the client connecting to Snowflake. This can help you to prevent unauthorized access from outside your organization.
- **VPC endpoints**: VPC endpoints allow you to connect to Snowflake from within a specific VPC. This can help to improve security by restricting access to your Snowflake account to only those users who are within your VPC.
- **Private links**: Private links allow you to connect to Snowflake from within a specific cloud provider's network. This can help to improve security by restricting access to your Snowflake account to only those users who are within your cloud provider's network.
- **TLS encryption**: All communication between Snowflake and your clients is encrypted using TLS. This helps to protect your data from unauthorized access while it is in transit.

Security policies

The following are the security policies:

- **Data encryption**: By default, the Snowflake encrypts all the data at rest and in transit using the industry-standard **AES 256-bit encryption** for robust protection. This helps to protect your data from unauthorized access even if your Snowflake account is compromised.
- **Multi-factor authentication (MFA)**: Snowflake supports MFA; it requires users to enter a code from their mobile device in addition to their password when they log in. This is an additional layer of security that prevents unauthorized access even if someone knows your password.
- **Password complexity requirements**: Snowflake requires passwords to be a minimum of 8 characters long and to contain a mix of upper and lowercase letters, numbers, and symbols. This helps to prevent unauthorized access by making it more difficult to guess passwords.

Best practices

In addition to the features and policies described above, there are many best practices that you can follow to help secure your Snowflake account and data. These best practices include:

- Only allow connections from known clients. You can use network policies to restrict access to your Snowflake account to only those clients that you trust.
- Use strong passwords and enable multi-factor authentication. This will help protect your account from unauthorized access.
- Limit access in line with the Least Privilege policy. Grant the minimum permissions that the user needs to do their jobs. This will reduce the risk of unauthorized access and prevent access to sensitive data.
- Review your access control settings regularly. This will help ensure that your account is secure and that only authorized users have access to data.

By following these best practices, you can help to keep your Snowflake account and data secure.

Single sign-on

Single sign-on (**SSO**) is a security feature that enables users to access one or more applications using a single set of credentials. This can simplify the login process and reduce the risk of unauthorized access.

Snowflake supports SSO using the **Security Assertion Markup Language** (**SAML**) 2.0 standard. This means that you can use your existing **identity provider** (**IdP**) to authenticate users to Snowflake. Once a user has authenticated to the IdP, they will be able to access Snowflake without having to enter their credentials again.

Multi-Factor Authentication

Multi-Factor Authentication (**MFA**) is a security feature that requires users to provide two or more pieces of evidence to authenticate themselves. This can help to reduce the risk of unauthorized access even if someone knows your password.

Snowflake supports MFA using the **Duo Security service**. When you enable MFA for a user, they will be prompted to approve Duo Push or enter a code from their Duo Mobile app in addition to their password when they log in. This helps to ensure that only authorized users can access your Snowflake account.

The following figure illustrates the MFA login flow of Snowflake:

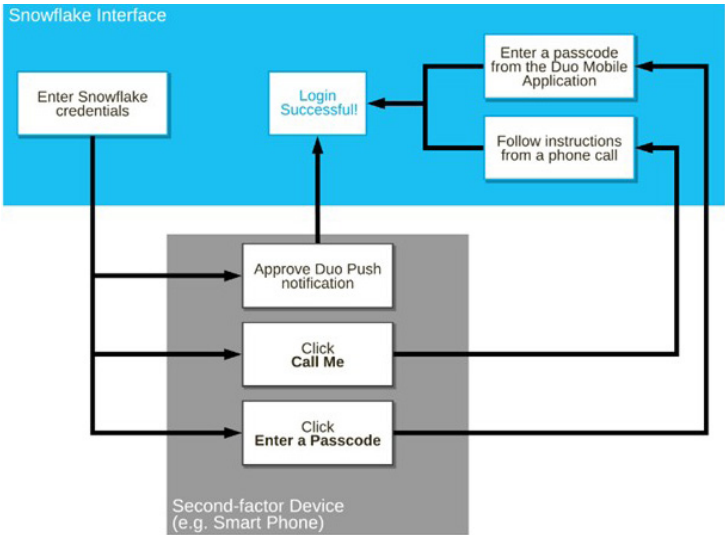

Figure 6.1: *Snowflake MFA Login Flow*

Using MFA with Snowsight

Snowflake Snowsight– login with MFA:

1. Point your browser at the URL for your account identifier. For example, **https://app.snowflake.com/**

   Enter your credentials (user login name and password). The following figure illustrates Snowflake login UI:

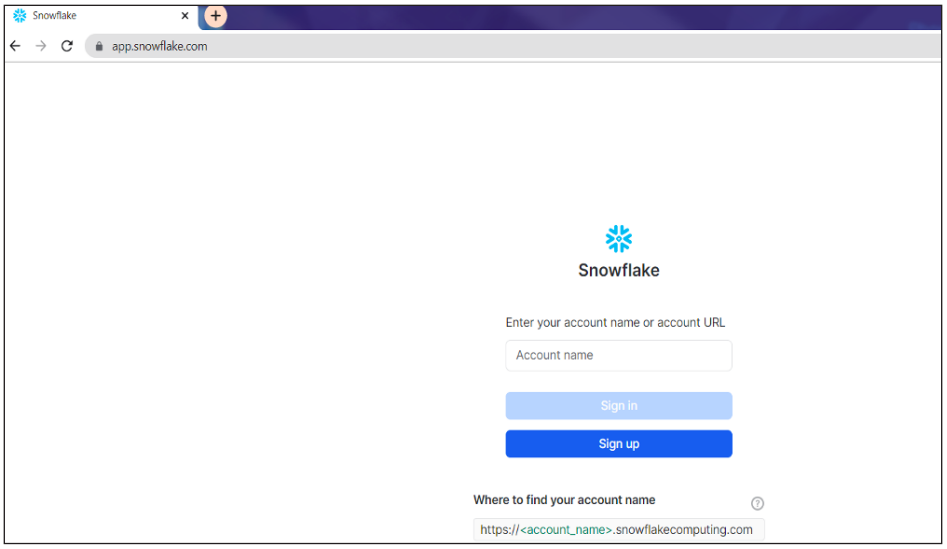

Figure 6.2: *Snowflake Snowsight Login*

2. Duo offers three convenient ways to verify your identity: push notification, phone call, or Passcode from the Duo app.
3. Let us say the user opts for the recommended preference as Duo Push, then a push notification will be sent to your Duo Mobile application. Simply click **Approve** when the user receives notification and will be automatically logged into Snowflake:

   Refer to the following figure for a sample Duo push notification to the users:

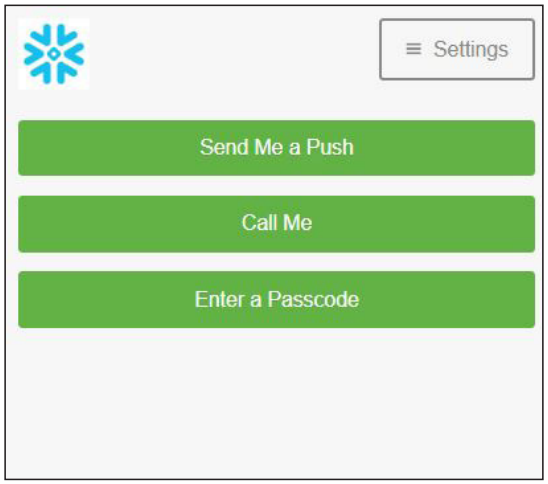

Figure 6.3: *Snowflake – Duo MFA*

Based on the above figure, the user can also choose other two options as part of the MFA authentication method to login to Snowflake:

1. Click **Call Me** to receive login instructions to the registered phone.
2. Click the **Enter a Passcode** option to log in by entering the passcode provided by the Duo Mobile application.

Benefits of using SSO and MFA

The use of SSO and MFA with Snowflake offers several benefits, including:

- **Simplified login process**: SSO can simplify the login process for users by allowing them to log in to multiple applications using a single set of credentials.
- **Reduced risk of unauthorized access**: MFA can help to reduce the risk of unauthorized access by requiring users to provide two or more pieces of evidence to authenticate themselves.
- **Improved security**: SSO and MFA can help to improve the security of your Snowflake account by making it more difficult for unauthorized users to access your data.

Configuring SSO and MFA

To configure SSO and MFA for Snowflake, you will need to follow the mentioned steps:

1. Create a SAML 2.0 security integration in Snowflake.
2. Configure your IdP to trust Snowflake.
3. Enable MFA for users in Snowflake.

For more information refer to the *Additional references* section for guidance and best practices on how to configure SSO and MFA in Snowflake.

Federated authentication in Snowflake

Federated authentication is a security feature that enables users to authenticate to Snowflake using their existing **identity provider** (**IdP**). This can simplify the login process and reduce the risk of unauthorized access.

Snowflake supports federated authentication using the SAML 2.0 standard. This means that you can use your existing IdP, such as Okta or Microsoft ADFS, to authenticate users to Snowflake. Once a user has authenticated to the IdP, they will be able to access Snowflake without having to enter their credentials again.

Supported identity providers

The following are the supported identity providers:

- Okta: Hosted service
- Microsoft AD FS: On-prem software
- Other SAML 2.0 compliant vendors as an idP include:
    - Google G Suite
    - Microsoft Azure Active Directory
    - OneLogin
    - Ping Identity PingOne

Benefits of federated authentication

There are several benefits to using federated authentication with Snowflake, including:

- **Simplified login process**: Federated authentication can simplify the login process for users by allowing them to log in to multiple applications using a single set of credentials. This can be especially beneficial for users who need to access Snowflake from multiple devices or locations.

- **Reduced risk of unauthorized access**: Federated authentication can help to reduce the risk of unauthorized access by requiring users to authenticate to their IdP before they can access Snowflake. This helps to ensure that only authorized users can access your data.
- **Improved security**: Federated authentication can help to improve the security of your Snowflake account by making it more difficult for unauthorized users to access your data. This is because the IdP is responsible for authenticating users and issuing tokens, which Snowflake then uses to verify user identity.

Configuring federated authentication

Federated authentication is a valuable security feature that can help simplify the login process and improve the security of your Snowflake account. If you are looking for a way to improve the security of your Snowflake account, then federated authentication is a good option to consider.

To configure federated authentication for Snowflake, you will need to:

1. Create a SAML 2.0 security integration in Snowflake.
2. Configure your IdP to trust Snowflake.
3. Configure your users to use federated authentication.

For more information refer to the *Additional references* section for guidance and best practices on how to configure federated authentication for Snowflake.

Conclusion

In this chapter, we learned about the Snowflake Security principles, how to set up the Snowflake Accounts access, and various network security policies to secure the account from unauthorized access. In addition, we also learned the details about the SSO, MFA, and Federated authentication of Snowflake and the benefits of these setups.

In the next chapter, we will learn about the Snowflake Roles, roles hierarchy, and technical details about Snowflake's data governance capabilities and how to secure data using Secure View, and so on.

Points to remember

Refer to the below summary for a quick reference to what we learned so far in this chapter:

- **What are the different available options to secure the data in Snowflake?**

  Snowflake security principles are designed to protect your data from unauthorized access, modification, or deletion. These principles are implemented using a variety of security controls, including:

- **Data encryption**: All data in Snowflake is encrypted at rest and in transit using industry-standard encryption algorithms.
    - **User authentication and authorization**: Users must be authenticated before they can access Snowflake. Authentication can be done using passwords, federated authentication, or OAuth. Once users are authenticated, they are granted access to specific resources based on their roles.
    - **Network security**: Snowflake provides several features to help secure your network, including network policies, private connectivity, and firewall rules.
        - **N**etwork policies allow you to control which IP addresses can access your Snowflake account.
        - Private connectivity allows you to connect to Snowflake from your on-premises network.
        - Firewall rules allow you to control which ports are accessible from the internet.
    - **Access control**: Snowflake provides a granular access control system that allows you to control who has access to specific resources. You can control access at the user, group, and role level.
    - **Auditing**: Snowflake provides a comprehensive auditing system that allows you to track all activity in your account. This includes who accessed what data, when they accessed it, and what they did with it.
    - **Compliance**: Snowflake is committed to compliance with a wide range of industry standards, including HIPAA, PCI DSS, and SOC 2. This means that Snowflake has implemented several security controls to meet the requirements of these standards.
- **List a few best practices for Setting Up Snowflake Account Access.**

  Following are a few best practices for Setting Up Snowflake Account Access:
    - **Designate at least two account administrators**. This will help your account always be backed up with at least one person who can perform account-level tasks, even if one of the account administrators is unable to log in.
    - **Use strong passwords and enable multi-factor authentication**. This will help protect your Snowflake account from unauthorized access.
    - **Grant the minimum permissions that the user needs to do their jobs**. This will reduce the risk of unauthorized access and prevent access to sensitive data.
    - **Review your access control settings regularly**. This will help ensure that your account is secure and that only authorized users have access to data.

- **How to Set Up Snowflake Account Access:**
    - Snowflake is a cloud-based data warehouse that provides secure and scalable access to data. To set up Snowflake account access, you will need to create an account, create users, grant roles to users, and configure access control.
- **The key concepts of Snowflake's Access Control Framework are:**
    - **Securable objects:** These are the objects that can be secured in Snowflake. Examples of securable objects include databases, schemas, tables, views, and functions.
    - **Roles:** These are the entities to which privileges can be granted. Roles are assigned to users, who can then use the privileges granted to the role to access securable objects.
    - **Privileges:** These define the level of access that a user has to a securable object. There are a variety of privileges that can be granted to a role, such as the ability to create, read, update, and delete data.
    - **Users:** These are the entities that are granted roles. Users can be individuals or groups of users.
- **What is Single Sign-On?**
    - SSO is a security feature that enables users to access one or more applications using a single set of credentials.
    - This can simplify the login process and reduce the risk of unauthorized access.
- **What is MFA and how Snowflake supports this setup?**
    - MFA is a security feature that requires users to provide two or more pieces of evidence to authenticate the login process.
    - Snowflake supports MFA using the **Duo Security service**. When you enable MFA for a user, they will be prompted to enter a code from their Duo Mobile app in addition to their password when they log in.
- **What is Federated Authentication and how Snowflake supports this setup?**
    - Snowflake supports federated authentication using the SAML 2.0 standard. This means that you can use your existing IdP, such as Okta or Microsoft ADFS, to authenticate users to Snowflake.
    - Once a user has authenticated to the IdP, they will be able to access Snowflake without having to enter their credentials again. This can simplify the login process and reduce the risk of unauthorized access.

- **List the Identity Providers that support Federated Authentication in Snowflake:**
    - Okta: Hosted Service
    - Microsoft AD FS: On-prem software
    - Other SAML 2.0 compliant vendors as an idP include:
        - Google G Suite
        - Microsoft Azure Active Directory
        - OneLogin
        - Ping Identity PingOne

Additional references

Refer to the following documentation to setup/configure the Snowflake Access control framework:

- Snowflake Documentation: Account Access: **https://docs.snowflake.com/en/user-guide/security-access-control.html**
- Snowflake Documentation: User Management: **https://docs.snowflake.com/en/user-guide/admin-user-management.html**
- Snowflake Documentation: Configuring Access Control: **https://docs.snowflake.com/en/user-guide/security-access-control-configure.html**

Refer to the following documentation to setup/configure Snowflake SSO and MFA authentication:

- Managing/Using Federated Authentication: **https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-use.html**
- Multi-Factor Authentication (MFA): **https://docs.snowflake.com/en/user-guide/security-mfa.html**

Refer to the following documentation to setup/configure Snowflake federated authentication:

- Configuring Snowflake to use Federated Authentication: **https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-configure-snowflake.html**
- Managing/Using Federated Authentication: **https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-use.html**

Chapter 7
Snowflake Security

Introduction

In this chapter, we will explore the core concepts of entities and roles, navigate the permission hierarchy, and unveil granular privilege control. Data governance takes center stage, showcasing robust capabilities. Finally, we will unlock the secrets of secure views, revealing how to safeguard sensitive data while maintaining its value.

Structure

In this chapter, we will discuss the following topics:

- Overview of Snowflake access control
- Snowflake entities/roles
- Roles hierarchy
- Roles and privilege management
- Data governance capabilities
- Secure views

Objectives

This chapter unlocks diverse mechanisms that protect your data in Snowflake. The readers will gain knowledge on various Snowflake access control techniques like roles, and their hierarchy, privilege management, data governance tools, and so on. In addition, readers will learn the power of Snowflake's various data governance capabilities to secure the data.

Overview of Snowflake access control

Snowflake's robust access control framework leverages a hybrid approach, combining **Role-Based Access Control** (**RBAC**) and **Discretionary Access Control** (**DAC**) to provide a granular and secure method for managing user permissions.

Following are the high-level summary of how the RBAC and DAC are used to secure the objects in Snowflake.

- RBAC establishes a foundation for efficient permission management. Permissions are assigned to predefined roles that align with business functions (e.g., data analyst, data scientist). These roles are then granted to individual users, streamlining administration, and ensuring adherence to the principle of least privilege.
- DAC empowers data owners with granular control over their data assets. They can grant specific access privileges to the RBAC roles, enabling them to tailor permissions based on user needs and data sensitivity. This fosters data ownership and accountability within the organization.

In Snowflake, access to securable objects is allowed via privileges assigned to roles, which are, in turn, assigned to users or other roles. Granting a role to another role creates a role hierarchy, which allows privileges to be inherited by child roles. In addition, each securable object has an owner that can grant access to other roles. The key concepts of Snowflake's Access Control Framework are:

- **Securable objects**: These are the objects that can be secured in Snowflake. Examples of securable objects include databases, schemas, tables, views, and functions.
- **Roles**: These are the entities to which privileges can be granted. Roles are assigned to users, who can then use the privileges granted to the role to access securable objects.
- **Privileges**: These define the level of access that a user has to a securable object. There are a variety of privileges that can be granted to a role, such as the ability to create, read, update, and delete data.
- **Users**: These are the entities that are granted roles. Users can be individuals or groups of users.

- **Object owner**: This is the user who created a securable object and has the most control over it.

By utilizing roles and privileges, administrators can precisely define who can access which objects and what actions they can perform, ensuring a high degree of flexibility and control over data access within the Snowflake environment.

The following figure illustrates how access control is managed within Snowflake:

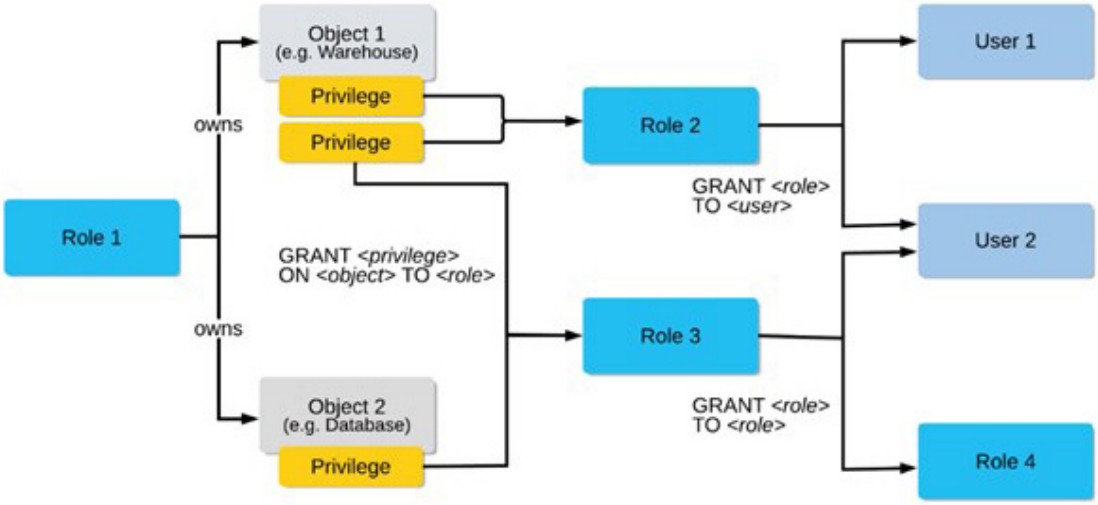

Figure 7.1: *Snowflake: Access Control (Credit: Snowflake documentation)*

Snowflake's Access Control framework offers a robust and scalable solution for managing data access. By providing granular control over permissions, RBAC enables administrators to grant users or roles the precise privileges necessary for their specific tasks, enhancing data security and mitigating the risk of unauthorized access.

For more information refer to the *Additional references* sections for best practices on how to set up the Snowflake access control framework.

Snowflake entities and roles

Snowflake uses a RBAC model to manage user access to data and resources. In RBAC, users are granted access to data and resources based on their assigned roles. The roles can be assigned to one or more users and vice versa, following are the details of how it is managed.

- **Users** are individuals or applications that have been granted access to Snowflake. Users are assigned roles, which define their privileges. A user can be assigned one or more roles. The privileges granted to a role determine what actions the user can perform in Snowflake.

- **Roles** are collections of privileges that can be assigned to users. Roles can be assigned to other roles, creating a role hierarchy. This allows you to grant users access to multiple securable objects by assigning them a single role.
- **Securable objects** are entities that can be granted access to, such as databases, schemas, tables, views, and stored procedures. When you grant a user access to a securable object, you are granting them the privileges that are associated with that object.

The following figure illustrates the list of objects and how it is organized within the hierarchy:

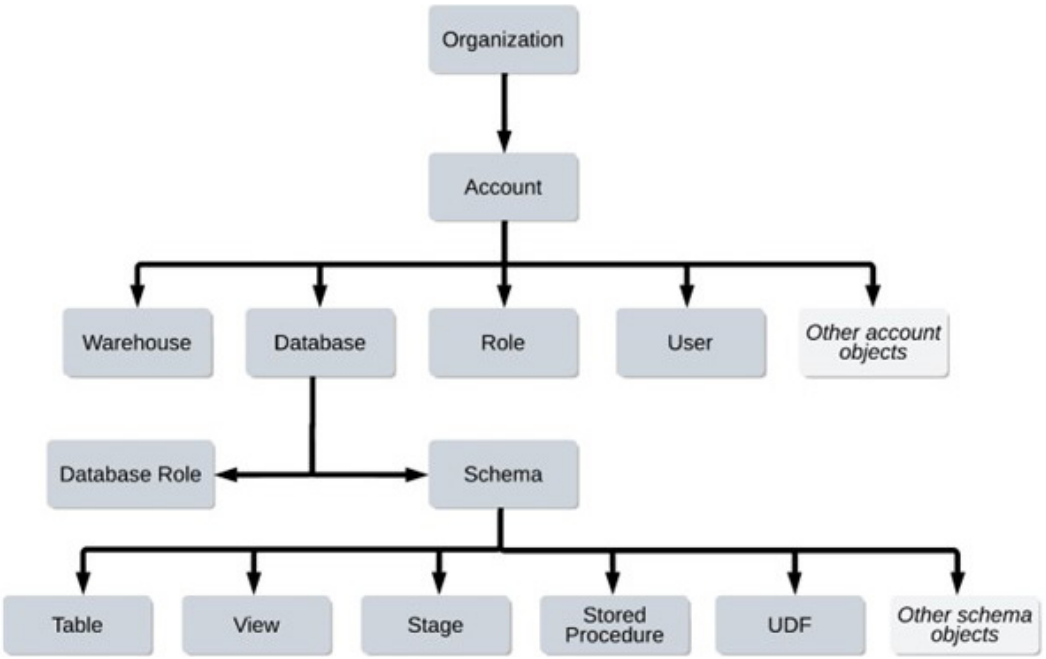

Figure 7.2: *Snowflake: Securable Objects Hierarchy (Credit: Snowflake documentation)*

The following are some of the key benefits of using roles in Snowflake:

- **Centralized management**: Roles can be managed centrally, which simplifies the administration of access control. This is because all of the privileges for a user are stored in a single role rather than being scattered across multiple objects.
- **Granularity**: Privileges can be granted to roles at a granular level, which allows for fine-grained control of access. This means that you can grant users the specific privileges that they need without giving them access to more than they need.
- **Flexibility**: Roles can be easily created, modified, and deleted, which makes it easy to adapt to changing business needs. This is because roles are not tied to any specific user or securable object.

Here are some examples of how roles can be used in Snowflake:

- A company might create a role called **Data Analyst** that has privileges to access all databases and schemas. This would allow data analysts to access all of the data in the company, which they need to do their jobs.
- A department might create a role called **Marketing Manager** that has privileges to access only the marketing database. This would allow marketing managers to access the data that they need to manage the marketing campaigns for the company.
- A user might create a role called **My Personal Role** that has privileges to access only the tables that they own. This would allow the user to control who has access to their data.

Following are some additional considerations:

- When creating roles, it is important to carefully consider the privileges that you need to grant. You should only grant the minimum privileges that are required to perform the tasks that the user needs to do.
- You should also consider the role hierarchy when granting privileges. If you grant a role a privilege, then all of the roles that are below it in the hierarchy will also have that privilege.
- It is important to regularly review the roles that you have created and make sure that they are still appropriate. You should also revoke privileges from roles that are no longer needed.

Roles hierarchy

A **role hierarchy** is a way of organizing roles in Snowflake so that they inherit privileges from each other. Roles are used to define the privileges that users have to databases, schemas, tables, and other securable objects. This can be a helpful way to manage access to Snowflake resources, as it allows you to grant privileges to a parent role and have those privileges automatically applied to all child roles.

To create a role hierarchy, you first need to create the parent role. Then, you can grant that role to other roles, which will become child roles. The child roles will inherit all of the privileges that are granted to the parent role:

- For example, you could create a parent role called **data access** and grant it the privilege to create and manage databases.
- Then, you could create child roles called **data analyst** and **data scientist** and grant those roles to users who need to access specific databases.
- The data analyst role would inherit the privilege to create and manage databases, but it would not have any privileges to create or manage tables or views.

- The data scientist role would inherit all of the privileges of the data analyst role, plus the privilege to create and manage tables and views.

By implementing this robust role hierarchy security model, organizations can significantly enhance the security posture of their Snowflake environment while ensuring that users have the appropriate access levels to fulfill their responsibilities. Here are some of the Key benefits of using role hierarchies in Snowflake:

- **Streamlined security administration**: Reduces the complexity of managing individual permissions.
- **Enhanced accuracy**: Ensures consistent and accurate inheritance of permissions, minimizing errors.
- **Optimized data access**: Provides users with appropriate access levels, promoting security best practices and enhancing productivity.

To effectively leverage role hierarchies, organizations should adhere to the following best practices while implementing the role hierarchy:

- **Least privilege principle**: Assign the highest-level permissions judiciously, granting them to selected individuals and not too many peoples in the organization.
- **Granular inheritance**: Customize inheritance settings for each role, specifying which permissions are inherited.
- **Cascading grants**: Utilize cascading grants to automatically propagate permissions to child roles when assigned to parent roles.
- **Regular audits**: Conduct periodic reviews of the role hierarchy to ensure alignment with evolving security requirements and organizational changes.

Roles and privilege management

Roles are collections of privileges, and privileges are permissions to perform specific actions on objects. In Snowflake, following are the two types of roles:

- **System defined roles** are predefined roles that come with certain privileges. They cannot be modified or deleted. These roles are designed to provide specific levels of access to Snowflake objects and data. For example, the **`ACCOUNTADMIN`** role has full control over the account, including the ability to create and manage users, roles, and databases. **`PUBLIC`** is a role that is granted to all users by default. It has very limited privileges, such as the ability to log in to Snowflake and view object metadata. This role is useful for users who need basic access to Snowflake, but do not need to perform any sensitive operations.
- **Custom roles** are roles that you create and define the privileges for. They can be assigned to users or other roles. This allows you to create roles that are specific

to the needs of your organization. For example, you could create a role for data analysts that allow them to query and analyze data but not to create or modify tables.

The type of role that you assign to a user will determine their level of access to Snowflake data and objects. It is important to carefully consider the tasks that the user will need to perform, the level of access that is required to perform those tasks, and the security risks associated with granting access to certain objects when creating or assigning roles.

Refer to the below section for the system-defined roles in Snowflake and how this can fit in various uses case scenarios with examples:

- **`ORGADMIN`**: This role has control over operations at the organization level. This includes creating and managing all the accounts in the organization, and viewing usage information across the organization.

  Example: You can assign this role to a corporate administrator who needs to managing all the accounts and viewing usage information across the organization.

- **`ACCOUNTADMIN`**: This role has full control over the account, including the ability to create and manage users, roles, and databases.

  Example: You can assign this role to a system administrator who needs to manage all aspects of the Snowflake account.

- **`SECURITYADMIN`**: This role can manage any object grant globally, as well as create,

  monitor, and manage users and roles.

  Example: You can assign this role to a security administrator who needs take care of all security implementation and security aspects/features.

- **`SYSADMIN`**: This role has the privileges needed to create databases, schemas and warehouses.

  Example: You can assign this role to an Architect or lead developer who needs to create and manage databases, schemas, and warehouses.

- **`USERADMIN`**: This role can create users and roles, but unlike **`SECURITYADMIN`**, can only affect the objects it has created.

  Example: You can assign this role to a database administrator who needs to create and manage users and roles for a specific database.

- **`PUBLIC`**: This role is granted to all users by default. It has very limited privileges, such as the ability to log in to Snowflake and view object metadata.

Here are some additional things to keep in mind about roles in Snowflake:

- Roles can be nested, which means that a role can inherit the privileges of another role. This allows you to create a hierarchy of roles that can be used to fine-tune access control.

- Roles can be granted to users or other roles. This allows you to delegate permissions to other users or roles.
- Roles can be revoked at any time. This allows you to quickly and easily remove access to data and objects.

Refer to the following figure for system defined roles, database and custom defined roles hierarchy within Snowflake:

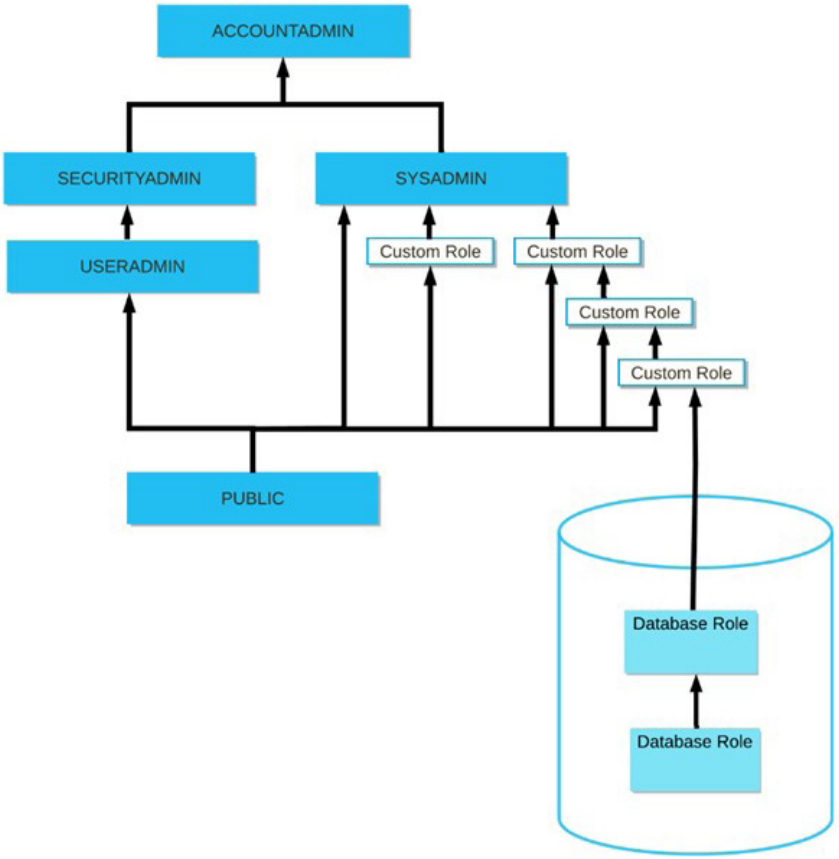

Figure 7.3: *Snowflake Role Hierarchy*

Note: `ORGADMIN` is a separate system role and it is not included in the hierarchy of system roles that manages operations at the organization level.

Data governance capabilities

Snowflake is a cloud-based data warehouse that offers a wide range of data governance capabilities. These capabilities help organizations to manage their data assets, protect them from unauthorized access, and ensure compliance with regulations.

Some of the key Snowflake data governance capabilities include:

- **RBAC**: It is a system that allows organizations to control who has access to their data and what they can do with it. Roles can be assigned to users, groups, or other entities. For example, an organization might create a role called **data analyst** that allows users to view and query data but not create or modify it.
- **Data masking**: Data masking is a technique that can be used to protect sensitive data from unauthorized access. Masking can be applied to individual columns or entire tables. For example, an organization might mask the Social Security numbers in its customer database so that they cannot be viewed by unauthorized users.
- **Data lineage**: Data lineage is the ability to track the movement of data through an organization. This can be helpful for understanding how data is used and for troubleshooting problems. For example, an organization might track the lineage of data used in a machine learning model to ensure that the model is using accurate and up-to-date data.
- **Auditing**: Auditing allows organizations to track who has accessed their data and what they have done with it. This can be helpful for compliance purposes and for detecting unauthorized access. For example, an organization might audit access to its financial data to ensure that it is being used for legitimate purposes.
- **Data quality**: Data quality is the degree to which data is complete, accurate, and consistent. Snowflake offers a number of tools to help organizations improve the quality of their data. For example, Snowflake offers a data profiling tool that can identify data quality issues.

These are just some of the data governance capabilities that Snowflake offers. Organizations can use these capabilities to meet their specific data governance needs.

In addition to the capabilities mentioned above, Snowflake also offers a number of other features that can help organizations with data governance, such as:

- **Row level security and column level security**: Both row level security and column level security can be used to protect sensitive data in Snowflake. This is a Snowflake Enterprise or higher edition feature.
- **Object tagging**: Object tagging allows organizations to tag their data assets with metadata, such as the data's sensitivity level or the business unit that owns it. This can be helpful for understanding and managing data assets.
- **Access history**: Snowflake's Access History feature allows you to track who has accessed your data and what they have done with it. This can be helpful for compliance purposes and for detecting unauthorized access.
- **Data classification**: Data classification is the process of assigning a data classification to each data asset. This can be helpful for understanding the sensitivity of data assets and for ensuring that they are protected appropriately.

- **Data governance policies**: Data governance policies are rules that organizations can define to govern how their data is managed. These policies can be used to enforce data quality, security, and compliance requirements.

Snowflake row level and column level security

Row-level and column-level security are vital features for safeguarding sensitive data within Snowflake. Row-level security governs access to specific rows within a table, while column-level security controls access to specific columns. Both mechanisms utilize policies based on factors such as user roles or table values to determine access privileges. This is a Snowflake Enterprise or higher edition feature.

Implementing row-level security through row access policies allows organizations to restrict access to specific rows based on predefined criteria. For example, a policy might allow only sales personnel to view customer data for a specific region. Similarly, column-level security, implemented through column masking policies, can obscure sensitive data such as credit card numbers from unauthorized users.

Here are some of the benefits of using row level security and column level security in Snowflake:

- **Protection of sensitive data**: Prevent unauthorized access to confidential information.
- **Regulatory compliance**: Ensure adherence to data protection regulations like GDPR.
- **Enhanced data quality**: Mitigate the risk of unauthorized modifications to critical data.
- **Streamlined data management**: Simplify access control and permissions management.

Both mechanisms are robust solutions for data protection within Snowflake. The specific approach (row-level or column-level security) will depend on the unique requirements and data sensitivity within each organization's specific requirements. By implementing appropriate security measures, Snowflake users can maintain data confidentiality and integrity while ensuring authorized personnel have the necessary access to perform their duties.

Object tagging

Object tagging in Snowflake provides a robust mechanism for classifying and organizing data assets. By assigning metadata tags to objects, organizations can enhance data governance, streamline compliance, and improve data discoverability. These tags can include details such as data sensitivity levels, ownership information, and lineage. This requires Enterprise Edition (or higher).

Leveraging object tagging, enterprises can establish granular control over their data landscape. This facilitates efficient data retrieval, ensures adherence to internal policies, and simplifies the process of meeting regulatory requirements.

Following are few key benefits of implementing object tagging in Snowflake:

- **Enhanced data discoverability**: Tags enable users to quickly identify and locate relevant data assets based on specific criteria.
- **Enforced data governance**: By incorporating tags into data governance policies, organizations can maintain data integrity and control access effectively.
- **Simplified compliance**: Tagging sensitive data ensures that appropriate safeguards are in place to meet industry and regulatory standards.

By implementing object tagging empowers organizations to optimize data management practices, ultimately enhancing data visibility, security, and compliance. This comprehensive approach aligns with industry best practices for data governance and facilitates seamless integration with existing data workflows.

Access history

Snowflake's access history feature allows you to track who, when, and how users access your data. This requires Enterprise Edition (or higher). The access history feature can be used to track the following information: **`Query_id`**, **`query_start_time`**, **`user_name`**, **`direct_objects_accessed`**, **`base_objects_accessed`**, **`objects_modified`**, **`object_modified_by_ddl`**, **`policies_referenced`**, **`parent_query_id`**, **`root_query_id`**, and so on.

For example, the following statement shows the access history for a specific user:

```
SQL :
-- Sample Syntax/Code to query ACCESS_HISTORY
SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.ACCESS_HISTORY
WHERE user_name like 'BK%'
;
```

The access history usage view can be a valuable option to review the access history of various objects. By tracking who has accessed your data and what they have done with it, you can help to ensure that your data is secure and that your organization is compliant with regulations.

Benefits of Snowflake data governance capabilities

Snowflake's robust data governance capabilities provide organizations with comprehensive tools to effectively manage and safeguard their information assets within the cloud data warehouse. By implementing a structured framework of rules and controls, Snowflake ensures data integrity, security, and accessibility.

Following are the key benefits implementing Snowflake's data governance features:

- **Data protection**: Safeguards sensitive information from unauthorized access and potential breaches through features such as data masking and row-level security.
- **Regulatory compliance**: Assists organizations in meeting industry standards and regulatory requirements for data privacy and security.
- **Data quality**: Ensures data accuracy, consistency, and completeness, facilitating reliable decision-making processes.
- **Granular access controls**: Implements flexible access policies to grant appropriate access to authorized users based on their roles and responsibilities, preventing unauthorized viewing or modification of data.
- **Efficient data organization**: Facilitates efficient data discovery and utilization through tagging and cataloging mechanisms.

By implementing Snowflake's data governance features, organizations establish a well-structured and secure environment for their data, fostering trust, promoting collaboration, and maximizing the value of their data-driven initiatives. In essence, Snowflake's data governance functions as a comprehensive solution to establish trust, maintain control, and maximize the value of data assets within the organization.

Secure views

Snowflake Secure Views offer a robust mechanism for granular control over data visibility within a Snowflake data warehouse. This ensures that sensitive information remains confidential while still facilitating necessary data access to the users. Secure views in Snowflake provide enhanced data protection and privacy compared to standard views, making them a valuable option for safeguarding sensitive information.

Here are key benefits of using secure views in Snowflake:

- **Enhanced data confidentiality**: Secure views ensure that only authorized personnel can access specific subsets of data, safeguarding sensitive information from unauthorized exposure. This is particularly crucial for industries with strict compliance requirements, such as healthcare and finance.

- **Simplified access management**: Administrators can efficiently manage permissions by creating multiple Secure Views tailored to different user groups or business functions. This streamlines access control processes and reduces administrative overhead.
- **Dynamic data masking**: Secure views can be configured to mask or obfuscate sensitive data, such as **Personally Identifiable Information** (**PII**), ensuring that it remains hidden from unauthorized individuals while still allowing them to access relevant non-sensitive information.
- **Protect view definitions**: The underlying query expression (view definition) of a standard view is visible to others, potentially exposing proprietary database structures. Secure views restrict access to the view definition, ensuring that only authorized users can view prevent gaining insights into the underlying data model/database structures.
- **Secure data sharing**: Safely share specific subsets of data with external partners or different departments within your organization. By leveraging secure views, organizations can confidently share data intended for them.

How secure views works?

Secure views are defined using standard SQL **`CREATE VIEW`** syntax with the addition of the **`SECURE`** keyword. Administrators specify the base tables, filtering conditions, and visible columns. Access to secure views is then granted to specific roles or users, ensuring that they only see the data intended for them.

Following are few real-world use cases utilizing Secure Views in Snowflake:

Consider a healthcare organization's patient records database. Secure views can be created to:

- **Limit clinician access**: Allow doctors and nurses to view relevant medical histories while restricting access to billing information.
- **Protect patient privacy**: Hide sensitive patient identifiers like **Social Security Numbers** (**SSNs**) from staff members who don't require them.
- **Facilitate research**: Provide researchers with de-identified data for analysis while adhering to privacy regulations.

Snowflake Secure Views are a cornerstone of data governance within the Snowflake ecosystem. If you are looking for ways to protect your data in Snowflake, secure views are a good option to consider. By enabling precise, role-based access to data, Snowflake Secure Views empower enterprises to derive maximum value from their data assets while maintaining the highest levels of security. Snowflake Secure Views empower organizations to uphold stringent security standards, mitigate data exposure risks, and ensure compliance with privacy regulations.

Conclusion

In this chapter, we learned about the Snowflake Roles, roles hierarchy, and different roles within the roles hierarchy to manage various privileges. We also learned about the technical details about Snowflake's data governance capabilities and how to secure data using the Secure Views as an extension of Snowflake's Security. Hope the chapter summary helps the readers to summarize overall learning and keynotes to remember for the exam.

In the next chapter, we will learn about the Snowflake virtual warehouse, various characteristics around the warehouse such as scaling policy, data cache, how to manage and monitor the warehouses, how the credit usage and billing works with the warehouse, and so on.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **How Access control framework is managed in Snowflake?**

  Snowflake's robust access control framework leverages a hybrid approach, combining RBAC and DAC to provide a granular and secure method for managing user permissions.

  - RBAC creates predefined roles that match different job functions, like data analyst or scientist. Users are assigned to these roles, making it easier to manage their access and ensure they only have the minimum permissions needed for their work.
  - DAC gives data owners the power to decide who can access their data and what they can do with it. This means they can tailor permissions for different roles based on how sensitive the data is and what the user needs to do. This approach makes data ownership clear and ensures people are responsible for the data they own.

- **Describe role hierarchy and key benefits of using the role hierarchy in Snowflake?**

  A role hierarchy is a way of organizing roles in Snowflake so that they inherit privileges from each other. Roles are used to define the privileges that users have to databases, schemas, tables, and other securable objects. This can be a helpful way to manage access to Snowflake resources, as it allows you to grant privileges to a parent role and have those privileges automatically applied to all child roles.

  Key advantages of leveraging role hierarchies in Snowflake include:

  - **Simplified security administration**: Streamlines the management of user permissions by consolidating them within roles.
  - **Improved accuracy**: Ensures consistent and accurate privilege inheritance, minimizing the risk of errors.

- **Optimized data access**: Provides users with appropriate access levels, aligning with the principle of least privilege and bolstering overall data security.

- **How Roles and privileges managed in Snowflake?**

  In Snowflake, user permissions are governed by roles, which are essentially collections of privileges that authorize specific actions within the system. There are two primary categories of roles:

  - **System-defined roles**: These are predefined roles available in Snowflake, each with a distinct set of privileges tailored for specific administrative functions. Modification or deletion of these roles is not permitted. Examples: **`ACCOUNTADMIN`**, **`PUBLIC`**, and so on.

  - **Custom roles**: These roles are created and configured by administrators to align with the specific needs of the organization. They allow for granular control over permissions and can be assigned to individual users or other roles. For instance, a custom role might be established for data analysts, enabling them to query and analyze data without the ability to modify database structures.

- **List the available Snowflake system defined roles and their functionalities.**

  These are predefined roles available in Snowflake, Modification or deletion of these roles is not permitted.

  - **`ACCOUNTADMIN`**: Possesses comprehensive control over the Snowflake account, including user and object management.

  - **`ORGADMIN`**: Oversees operations at the organization level, managing multiple accounts and accessing usage insights.

  - **`SECURITYADMIN`**: Responsible for managing security implementations, grants, and access controls.

  - **`USERADMIN`**: Creates and manages users and roles, with permissions limited to the objects they create.

  - **`SYSADMIN`**: Empowered to create and manage core database objects like databases, schemas, and warehouses.

  - **`PUBLIC`**: The default role assigned to all users with minimal privileges, primarily for login and metadata viewing.

- **What is Object tagging in Snowflake and list few key benefits?**

  Object tagging in Snowflake is a metadata management feature that allows you to classify and organize your data assets. By assigning tags to tables, views, and columns, you can add relevant information such as data sensitivity level, ownership, or lineage.

Key benefits include:

- **Easier data discovery**: Find the data you need more quickly.
- **Stronger data governance**: Enforce data policies consistently.
- **Simpler compliance**: Easily track and report compliance information.

- **Describe Snowflake's row and column level security feature.**

  Row-level and column-level security are essential mechanisms for safeguarding sensitive data within Snowflake. Row-level security governs access to specific rows within a table based on predefined criteria, such as user roles or table values. Column-level security, on the other hand, controls access to specific columns, often through masking sensitive data like credit card numbers.

- **What is Snowflake Secure view and key benefits of using the secure views?**

  Secure Views are a type of view available in Snowflake provides enhanced data protection and privacy compared to standard views, making them a valuable option for safeguarding sensitive information. Secure views are a good way to restrict access to sensitive data, such as financial data or customer data.

  Key benefits of using Snowflake Secure Views:

  - **Stronger data privacy**: It hides the underlying structure (tables, columns, etc.) and logic (SQL query) used to create the view. The secure view definition is not visible all the users and only exposed to authorized users.
  - **Safe data sharing**: Safely share specific subsets of data without exposing the entire database.
  - **Data masking**: Apply masking techniques to further protect sensitive information.
  - **Easier data governance**: Enforce consistent data governance policies and comply with data privacy regulations more easily.

Additional references

Refer to the following documentation to setup/configure the Snowflake Access control framework and RBAC:

- **Access control overview**: **https://docs.snowflake.com/en/user-guide/security-access-control-overview.html**
- **Account access**: **https://docs.snowflake.com/en/user-guide/security-access-control.html**
- **User management**: **https://docs.snowflake.com/en/user-guide/admin-user-management.html**
- **Configuring access control**: **https://docs.snowflake.com/en/user-guide/security-access-control-configure.html**

Chapter 8
Snowflake Virtual Warehouse and Warehouse Management

Introduction

This chapter discusses the details of Snowflake's virtual warehouses, equipping you with the knowledge to optimize performance, maximize efficiency, and effectively manage costs. We will comprehensively explore:

- **Foundational principles**: Warehouse types, configurations, and robust scaling policies.
- **In-depth technical considerations**: Data caching mechanisms, warehouse management and monitoring techniques, and comprehensive credit usage and billing insights.
- **Advanced strategies**: Leveraging auto-suspend/resume functionality, fine-tuning configurations for optimal performance, and implementing best practices for efficient data warehousing.

Structure

In this chapter, we will cover the following topics:

- Virtual warehouse overview
- Warehouse characteristics, and configurations
- Scaling policy

- Using the data cache
- Management/monitoring
- Credit usage and billing
- Warehouse settings and access

Objectives

This chapter aims to provide comprehensive information about Snowflake virtual warehouses, various characteristics and configurations around the warehouse, scaling policy, data cache, how to manage and monitor the warehouses, and the credit usage and billing work with the warehouse. By the end of this chapter, readers will gain knowledge on how to manage Snowflake Warehouses and be capable of ensuring efficient query execution, maximizing resource utilization, achieving cost-effective data warehousing, and so on.

Virtual Warehouse overview

A Snowflake virtual warehouse is a cluster of compute resources that are used to process queries and perform other DML operations. It is a fundamental concept in Snowflake's architecture, as it is the unit of compute that is provisioned and billed.

Here are some of the benefits of using Snowflake virtual warehouses:

- **Scalability**: Virtual warehouses can be scaled up or down automatically to meet the needs of your workload. This means that you only pay for the resources that you use.
- **Performance**: Snowflake virtual warehouses are designed to deliver high performance for even the most demanding workloads.
- **Flexibility**: Virtual warehouses can be used to run a variety of workloads, including data warehousing, data lakes, and data science.
- **Security**: Snowflake offers a comprehensive set of security features to protect your data.

Warehouse characteristics and configurations

Virtual warehouses are available in two types: **Standard** and **Snowpark-optimized**:

- **Standard virtual warehouses** are the default type of warehouse and are a good choice for most workloads. They offer a variety of sizes to choose from, so you can select the one that best meets your needs.
- **Snowpark-optimized** virtual warehouses are designed for running Snowpark applications. Snowpark is a framework that allows you to run Python and Scala

code in Snowflake. These warehouses offer more memory and CPU resources than standard warehouses, so they are well-suited for running complex data science and machine learning workloads.

Each virtual warehouse has a size, which determines the number of compute resources that are allocated to it. These are called as t-shirt sizes and not all warehouses have these available sizes that are dependent on the type of warehouse - Standard or Snowpark optimized. The available sizes are:

- X-Small
- Small
- Medium
- Large
- X-Large
- 2X Large
- 3X Large
- 4X Large
- 5X Large
- 6X Large

The size of the virtual warehouse that you need will depend on the workload that you are running. Refer to the below example for various use cases and expected warehouse allocations:

- **X-Small**: This is the smallest size virtual warehouse. It is suitable for small workloads, such as running a few queries per day.
- **Small**: This is a good size virtual warehouse for most workloads. It can handle a moderate number of queries per day.
- **Medium**: This is a larger size virtual warehouse that can handle a high number of queries per day.
- **Large**: This is a large size virtual warehouse and suitable more demanding workloads.
- **X-Large**: These are the X-Large Warehouses ranging from 1X to 6X-Large suitable for very demanding workloads, such as running large-scale data analysis or machine learning models.

Refer to the following chart for the comparison between Standard and Snowpark optimized warehouses and characteristics of the individual warehouse, available sizes and credits calculations:

	Standard Warehouse		Snowpark Optimized Warehouse		
Warehouse Size	**Availability**	**Credits/ Hour**	**Availability**	**Credits/Hour**	**Notes**
X-Small	✓	1	NA	NA	Default size for warehouses created in Snowsight and using `CREATE WAREHOUSE` command
Small	✓	2	NA	NA	
Medium	✓	4	✓	6	
Large	✓	8	✓	12	Default size for warehouses created using the Classic Console
X-Large	✓	16	✓	24	
2X-Large	✓	32	✓	48	
3X-Large	✓	64	✓	96	
4X-Large	✓	128	✓	192	
5X-Large	✓	256	✓	384	Generally available in AWS and Microsoft Azure regions, and in preview in US Govt. regions.
6X-Large	✓	512	✓	768	

Table 8.1: *Snowflake – Available warehouse charts*

The virtual warehouses can be created and managed using the Snowflake web interface or by SQL command. When you create a virtual warehouse, you can specify the following options:

- **Name**: The name of the virtual warehouse.
- **Type**: Standard or Snowpark-optimized Warehouses.
- **Size**: The size of the virtual warehouse determines the number of compute resources that are allocated to it.

Following are the advanced options to the Snowflake warehouse setup:

- **Auto-suspend and auto-resume settings**: These settings control whether the virtual warehouse is automatically suspended when it is idle and automatically resumed when it is needed.

- **Standalone or multi-clustered warehouses**: The standalone is a single cluster Warehouse; Multi cluster is an Enterprise and above edition features that has multi nodes with min and maximum nodes in a cluster to meet the larger workloads.

  You can also specify the following options for multi-cluster virtual warehouses:

  - **Minimum number of clusters:** The minimum number of clusters (1 to 10) that the virtual warehouse can have.
  - **Maximum number of clusters:** The maximum number of clusters (1 to 10) that the virtual warehouse can have.
- **Scaling policy**: Standard or Economy to scale to meet the demand and workload.
- **Query acceleration**: This setup accelerate outlier queries with additional flexible compute resources.

Virtual warehouses can be resized at any time. When you resize a virtual warehouse, the new size will take effect immediately for all new and queued queries. By default they are charged minimum one minute then subsequent per-second basis on new resized warehouse. The cost will depend on the size of the virtual warehouse and the amount of time that it is running.

Refer to the below figures for the various options and characteristics available when users try to create Standard virtual warehouses (type/size, and so on) in Snowflake:

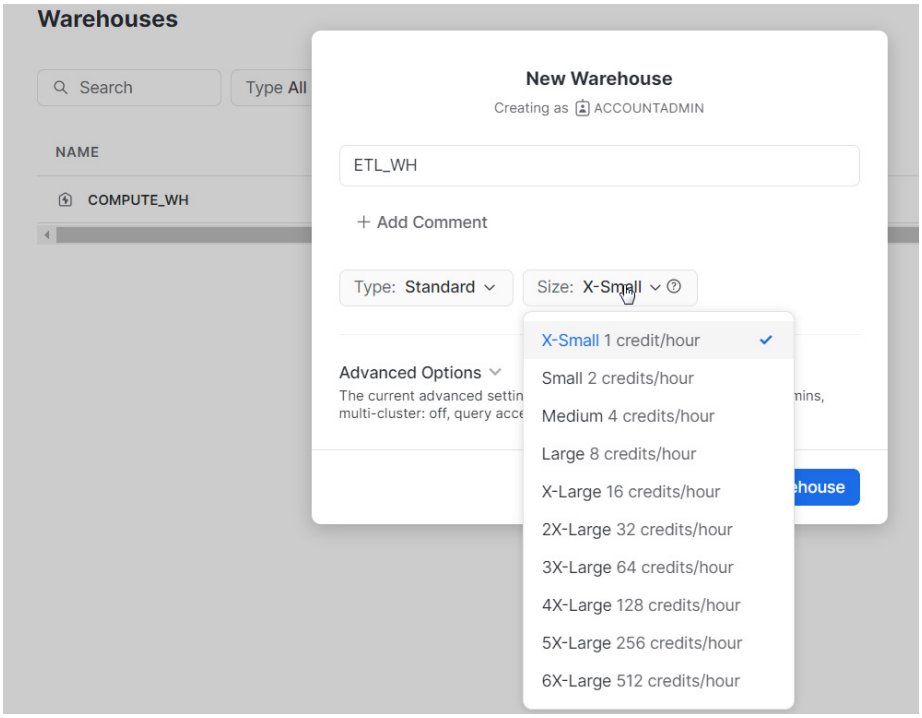

Figure 8.1 *Snowflake Standard Warehouses*

Refer to the following figures for the various options and characteristics available when users try to create Snowpark-optimized virtual warehouses (type/size, and so on) in Snowflake.

Note: The X-small and small warehouses are not available in Snowpark optimized warehouse type.

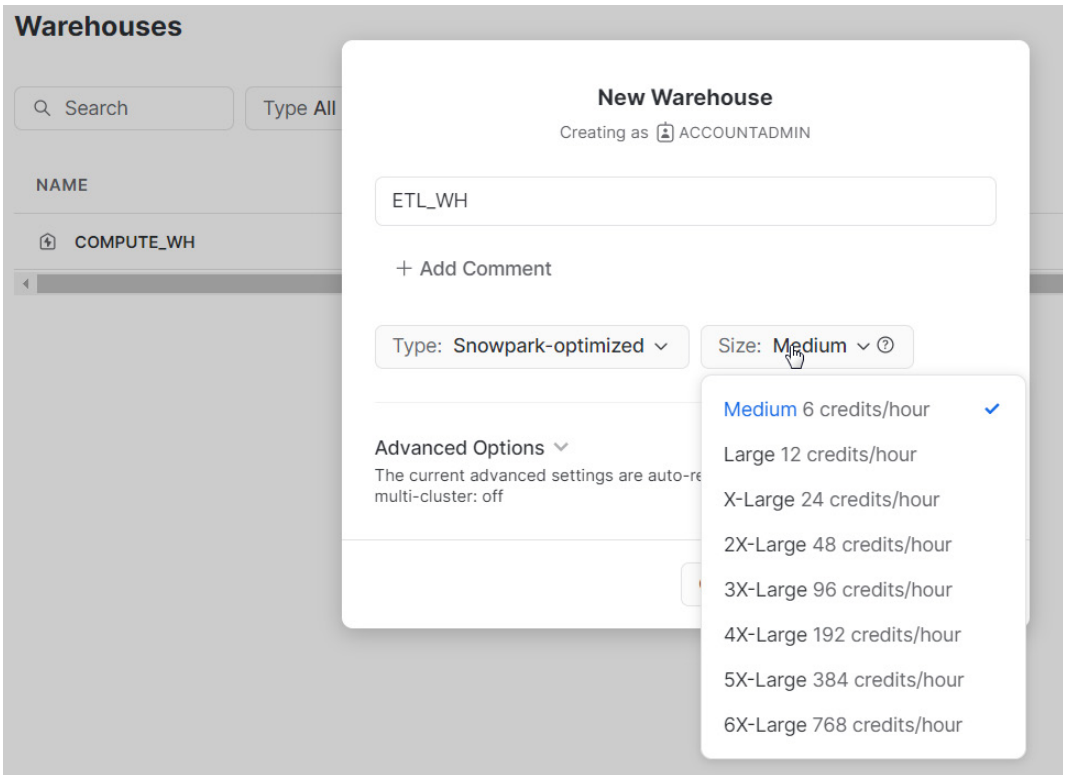

Figure 8.2: *Snowflake Snowpark Optimized Warehouses*

Refer to the following figures for the various advanced options and characteristics available related to auto resume, auto suspend, min & max multi-cluster setup, and scaling policy of the virtual warehouses in Snowflake.

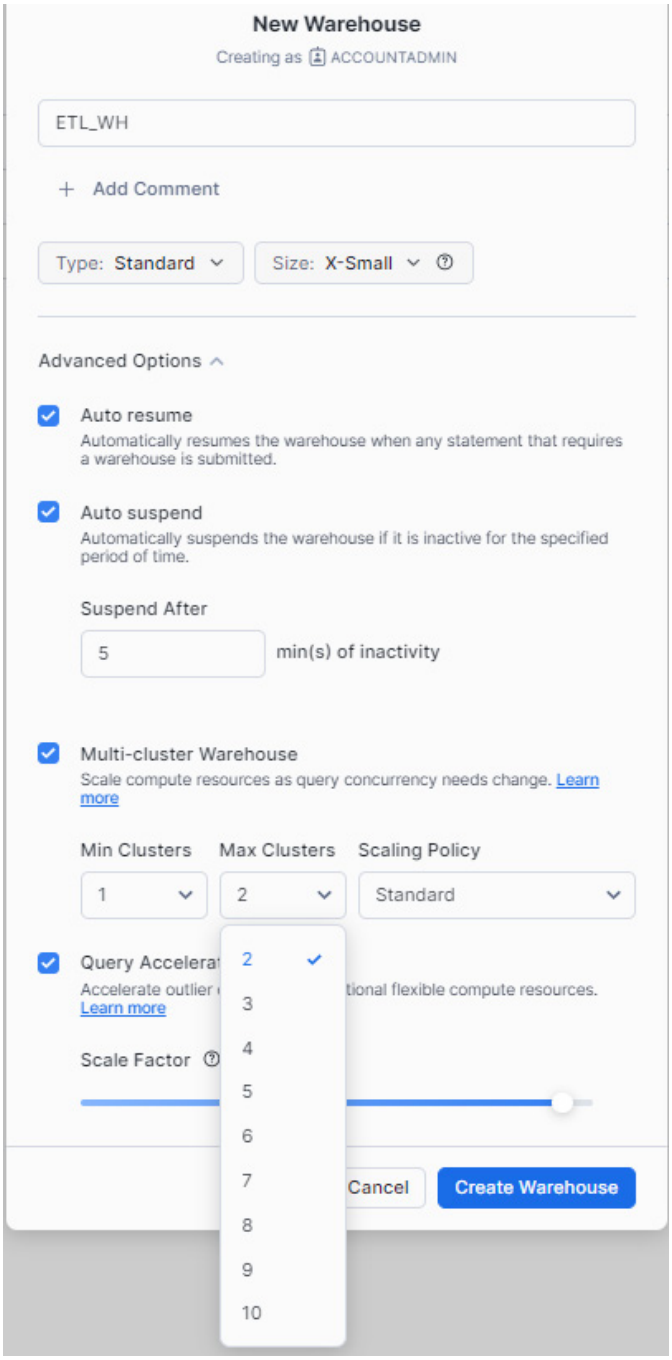

Figure 8.3: *Snowflake Create Warehouse – Advanced Options*

In addition to the benefits mentioned above, Snowflake virtual warehouses also offer the following features:

- **Multi-cluster support**: Virtual warehouses can be configured to use multiple clusters, which means they can be spread across multiple Snowflake compute clusters. This can help improve performance for workloads that require a lot of compute resources or high concurrency.

  When a virtual warehouse is configured as a multi-cluster, you can specify the minimum and maximum number of clusters that the virtual warehouse should have. This can help to ensure that the virtual warehouse has enough compute resources to meet the needs of the workload, while also avoiding over-provisioning.

- **Auto-scaling**: Virtual warehouses can be configured to scale up or down automatically based on workload demand. This can help you save money by only paying for the resources that you use.

- **Auto-suspend and auto-resume**: Virtual warehouses can be configured to automatically suspend when they are idle and automatically resume when they are needed. This can help to save costs by preventing virtual warehouses from running when they are not in use.

- **Cost-effectiveness**: Snowflake virtual warehouses are a cost-effective way to run a variety of workloads. You only pay for the resources that you use, and there are no upfront costs or commitments.

When choosing the characteristics and configurations for a Snowflake virtual warehouse, you should consider the following factors:

- The type of workload that you will be running.
- The amount of data that you will be processing.
- The number of concurrent users that you plan to use.
- The performance requirements of your workload and projected cost.

Scaling policy

Snowflake offers a robust and flexible scaling model, enabling organizations to dynamically adjust compute resources to meet varying workload demands. Snowflake Scaling Policy is a mechanism that determines when to start or shut down a cluster in a multi-cluster warehouse running in Auto-scale mode. The policy is designed to balance the need to prevent query queuing with the need to conserve credits. The scaling policy is set using the **`SCALING_POLICY`** property, and the following values are available:

- **`STANDARD`**: The default scaling policy is **Standard**, which prioritizes preventing query queuing over conserving credits. This means that additional clusters will be started, even if they are not fully utilized, in order to ensure that queries do not queue. It prevents or minimizes queuing of queries by favoring starting additional

clusters over conserving credits. The WH can be shut down based on the 5 to 6 consecutive checks performed at 1-minute intervals to determine whether the least loaded cluster workloads can be redistributed to other clusters without spinning the cluster again.

- **`ECONOMY`**: This policy conserves credits by starting additional clusters only when there are enough queries queued to justify the cost. This means that additional clusters will only be started when there are enough queries queued to keep the clusters busy for at least 6 minutes. The WH can be shut down based on the 5 to 6 consecutive checks performed at 1-minute intervals to determine whether the least loaded cluster workloads can be redistributed to other clusters without spinning the cluster again.

The scaling policy can be changed at any time. However, it is important to note that changing the scaling policy can have a momentary impact on performance, as Snowflake needs to stop and start clusters in order to implement the new policy.

To determine which scaling policy is right for your workload, you need to consider the following factors:

- **Expected workload**: If the workload is expected to be high, then the Standard policy should be used to prevent query queuing.
- **Cost**: The Economy policy can save credits, but it may result in some queries being queued.
- **Acceptable level of performance degradation**: If some performance degradation is acceptable, then the Economy policy can be used to save credits.

Ultimately, the best way to choose a scaling policy is to experiment with different policies and see what works best for the specific workload.

Snowflake virtual warehouse: horizontal and vertical scaling

Snowflake's virtual warehouses offer on-demand elastic scaling through a combination of vertical and horizontal scaling mechanisms. This is different from traditional data warehouses where you're stuck with a fixed amount of power. Snowflake lets you scale your data processing power up or down dynamically and adjust compute resources based on workload requirements. Vertical scaling allows for increasing the processing power of individual warehouses, ideal for handling complex queries. Horizontal scaling provides the ability to spin up additional warehouses, effectively distributing workloads across a larger pool of resources. This combined approach ensures optimal performance for varying workloads while optimizing costs by paying only for the utilized resources. Following are few additional information along with real time use cases.

Vertical scaling: scale up or down

Vertical scaling involves adjusting the size of an individual virtual warehouse. Larger warehouse sizes offer increased compute power (CPU and memory) to handle complex queries and larger datasets. Snowflake provides a range of warehouse sizes, from X-Small to 6X-Large, allowing users to select the appropriate size based on workload requirements.

Increasing the warehouse size (scaling up) allocates additional resources, enhancing performance for complex queries or large datasets. Conversely, scaling down can be employed during periods of lower demand to reduce costs. Vertical scaling or resizing a warehouse can be performed on-demand (manually), even while queries are executing, allowing for seamless adjustments.

Example use case: A business intelligence team running complex analytical queries on a large data warehouse may choose to vertically scale up their warehouse during peak reporting periods to ensure timely query execution. Conversely, they may downscale the warehouse during off-peak hours to minimize costs.

Horizontal scaling (auto-scaling) – scale out

Horizontal scaling leverages Snowflake's multi-cluster warehouse architecture. Horizontal scaling, also known as auto-scaling, dynamically adjusts the number of virtual warehouses (clusters) running in parallel to meet fluctuating workload demands. When query concurrency increases, additional clusters are automatically provisioned to share the workload, ensuring consistent performance. As demand subsides, clusters are gracefully suspended to optimize resource utilization and reduce costs. Snowflake offers two modes for multi-cluster warehouses:

- **Auto-scale mode**: Snowflake autonomously manages cluster provisioning and suspension based on real-time workload analysis. This mode is enabled by specifying min and max cluster to a different value.
- **Maximized mode**: Users define a fixed number of clusters to ensure maximum capacity is available, irrespective of workload fluctuations. This mode is enabled by specifying min and max clusters to the same value(must be >1).

Example use case: An e-commerce platform experiencing a sudden spike in website traffic during a promotional event can rely on auto-scaling to seamlessly handle the increased query volume without sacrificing website responsiveness.

The following figure shows the scaling policy options available for users to select when we create new Virtual Warehouses in Snowflake:

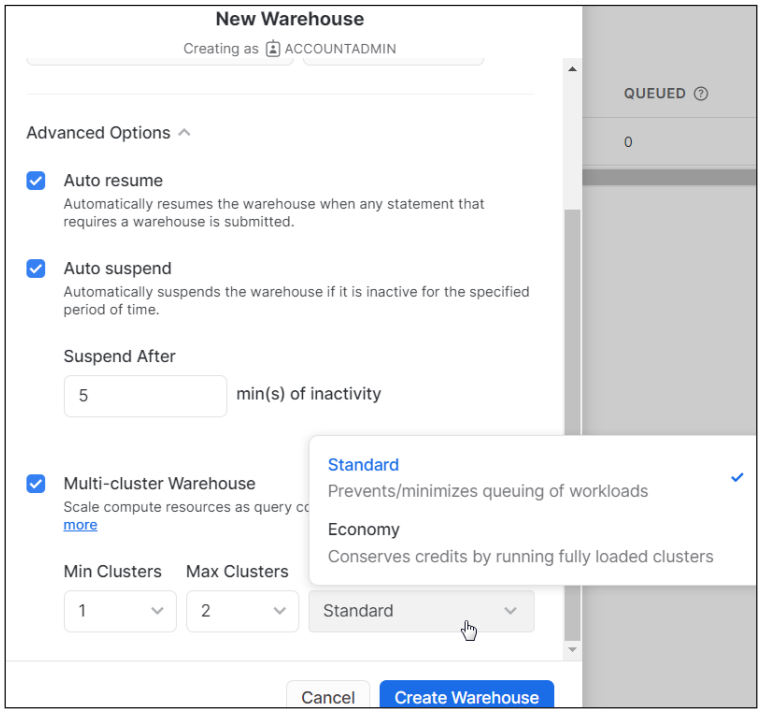

Figure 8.4: *Snowflake Scaling Policies*

Using the data cache

Snowflake's caching mechanism is a multi-layered architecture designed to optimize query performance, resource utilization and enhance the overall user experience. This multi-tiered system comprises three distinct caches:

- **Query result cache**: This persistent cache retains query results for 24 hours, allowing subsequent identical queries to be fulfilled directly from the cache, thus eliminating the need for recomputation. The Query Result Cache is shared across all users, however users cannot share the query results of other users, and this is maintained at the Cloud Service Layer.
- **Metadata cache**: By storing metadata such as table schemas and column types, this cache minimizes the latency associated with retrieving this information from disk, accelerating query planning and optimization. This information is used by Snowflake to accelerating query planning and optimizes the query. The Metadata Cache is also shared across all users and warehouses. The Virtual Warehouse Cache is specific to each warehouse, so the data that is cached in one warehouse is not available to other warehouses. Note Suspending the Warehouse will delete this cache.
- **Virtual warehouse cache**: This cache, specific to each virtual warehouse, stores frequently accessed data in memory for faster retrieval compared to disk storage.

The Virtual Warehouse Cache is specific to each warehouse, so the data that is cached in one warehouse is not available to other warehouses. Note Suspending the Warehouse will delete this cache.

The query result and metadata caches are shared across all users and warehouses, enhancing the overall efficiency of the system. Snowflake's caching strategy significantly improves query performance, particularly for those that are executed frequently or involve large datasets. Additionally, Snowflake employs other optimization techniques, such as:

- **Persisted query results**: Enabling reusability of query results for subsequent queries, even with underlying data changes.
- **Result set streaming**: Improving performance for queries returning large amounts of data by streaming results to the user as they are generated.
- **Columnar storage**: Optimizing queries accessing a limited number of columns by storing data in columns rather than rows.

This comprehensive caching architecture plays a pivotal role in optimizing Snowflake's performance, efficiency, and user experience. By strategically caching data and query results, Snowflake ensures that users can extract insights from their data with minimal latency and resource consumption, resulting in:

- **Accelerated query performance**: Significantly reducing query latency, particularly for recurring or complex queries.
- **Optimized resource utilization**: Eliminating redundant computations and minimizing remote data retrieval, resulting in cost reduction.
- **Enhanced user experience**: Providing a more responsive and interactive experience through faster query responses.

In conclusion, Snowflake's caching mechanism plays a pivotal role in achieving high performance, efficiency, and user satisfaction within the platform.

Management/monitoring

Warehouse management in Snowflake is the process of creating, managing, and monitoring warehouses. Warehouses are the compute resources that are used to run queries and other operations in Snowflake.

Here are some best practices for warehouse management:

- **Monitor your warehouses regularly**: Monitoring your warehouses can help you to track their usage and performance. This information can help you to optimize your warehouse usage and to identify any potential problems.
- **Use the right size warehouse for your workload**: The size of the warehouse should be appropriate for the amount of compute resources that you need. Using a larger warehouse than you need can waste credits, and using a smaller warehouse than

you need can affect performance. Continue monitor the workload and readjust the warehouse size as necessary.

- **Enable and set the auto-suspend time appropriately**: The auto-suspend time determines how long a warehouse will remain running after it has been idle. Most cases it's not necessary the warehouse continue to run if there are no demands. Enable and adjust the auto-suspend setting according to your workload requirements.
- **Isolate the workload by provisioning dedicated resource**: Snowflake enhances operational efficiency and scalability by leveraging a multi-cluster architecture to implement workload isolation. This approach involves assigning dedicated virtual warehouses to distinct business functions such as ETL, reporting, and analytical queries. By segregating workloads, Snowflake ensures optimal resource allocation, preventing resource contention and guaranteeing consistent performance across diverse business functions. This strategic isolation of workloads not only optimizes current operations but also enables seamless adaptability to meet evolving business requirements.
- **Use the right type of warehouse for your workload**: The type of warehouse determines the features that are enabled. For example, Snowpark-optimized warehouses are designed for use with Snowpark, which is a framework for running Python and Scala code in Snowflake.
- **Use efficient queries**: Queries that are inefficient can consume more credits than efficient queries. Fine tune and optimize the queries for better performance.

Credit usage and billing

Snowflake compute cost is priced/billed on a per-second basis, with a minimum of 60 seconds whenever the warehouse starts or resumes based on the hourly rate associated with the account. After one minute, all the subsequent billing is per second as long the warehouse runs continuously. Also, suspending and then resuming or resizing the warehouse will follow a similar minimum 60-second billing. Overall, the compute cost or credits are calculated for the following activities related to any query or data processing to Snowflake.

For the virtual warehouse, the total cost will be calculated based on the warehouse size (X-small to 6X large), the number of virtual warehouses, and how long they run to complete the activity. Following are the subsections of the compute cost model:

- **Cloud service compute**: It is completely managed by Snowflake, and the cost is involved for various cloud service layer activities.
- **Virtual warehouse compute**: This is managed by the user, so the users have the flexibility to control/manage how the virtual warehouse credits will be utilized to perform query execution, load data, and perform any DML operation. Snowflake's virtual warehouses offer on-demand elastic scaling through a combination of

vertical and horizontal scaling mechanisms. This combined approach ensures optimal performance for varying workloads while optimizing costs by paying only for the utilized resources.

- **Serverless compute**: The Serverless features are completely managed by Snowflake. Serverless compute cost is calculated based on the total usage of Snowflake managed to compute measured in compute hours. The compute hours are calculated on a per-second basis and rounded up to the nearest whole second.

User can track their credit usage and billing in Snowsight, the Snowflake web interface. Snowsight provides a variety of reports that you can use to track your usage and costs. This can help you to identify areas where you can optimize your usage and save credits.

Conclusion

In this chapter, we learned about the overview of the Snowflake virtual warehouse, various characteristics and configurations around the warehouse, scaling policy etc. In addition, we also learned about other characteristics of the warehouse, such as scaling policy, data cache, how to manage and monitor the warehouses, how the credit usage and billing works with the warehouse. Hope the chapter summary helps the readers to summarize overall learning and keynotes to remember for the exam.

In the next chapter, we will learn about various performance characteristics of Snowflake that includes the Query profile, Query history, and characteristics of Query history, how to analyze and improve Snowflake query performance using various commands and techniques.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **What is virtual Warehouse in Snowflake and key benefits?**

  A Snowflake virtual warehouse is a cluster of computing resources that can process queries and execute **Data Manipulation Language** (**DML**) operations. It is a core component of Snowflake's architecture, serving as the billable unit of compute power.

  Key benefits of leveraging Snowflake virtual warehouses include:

  - **Scalability**: Virtual warehouses can dynamically scale to accommodate fluctuating workloads, ensuring optimal resource utilization and cost efficiency.
  - **Performance**: Snowflake's virtual warehouses are engineered to deliver high performance for even the most demanding data processing tasks.
  - **Versatility**: These warehouses can handle a variety of workloads, spanning data warehousing, data lakes, and data science applications.
  - **Cost-effectiveness**: Warehouses can be suspended when not in use, eliminating unnecessary costs.
  - **Elasticity**: They can be provisioned quickly, adapting to changing needs.

- **List the available types of virtual warehouses in Snowflake:**
    - **Standard warehouses**: These are the general-purpose warehouses suitable for most workloads, including various SQL queries and data processing tasks.
    - **Snowpark-optimized warehouses**: These virtual warehouses are designed for running Snowpark applications. Snowpark is a framework that allows you to run Python and Scala code in Snowflake.
- **List the available virtual warehouse size in Snowflake:**

  Refer to the following chart for the comparison between Standard and Snowpark optimized warehouses and characteristics of the individual warehouse, available sizes and credits calculations:

Type of Virtual Warehouse/ # of Clusters	X-Small	Small	Medium	Large	X-Large	2X-Large	3X-Large	4X-Large	5X-Large	6X-Large
Standard	1	2	4	8	16	32	64	128	256	512
Snowpark-optimized	n/a	n/a	6	12	24	48	96	192	384	768

Table 8.2: *Snowflake –Warehouse Chart*

- **What is the pricing model used for Snowflake Virtual Warehouses to calculate the cost?**

  Snowflake compute cost is priced/billed on a per-second basis, with a minimum of 60 seconds whenever the warehouse starts or resumes based on the hourly rate associated with the account. After one minute, all the subsequent billing is per second as long the warehouse runs continuously. Also, suspending and then resuming or resizing the warehouse will follow a similar minimum 60-second billing. Overall, the compute cost or credits are calculated for the following activities related to any query or data processing to Snowflake.

  For the virtual warehouse, the total cost will be calculated based on the warehouse size (X-small to 6X large), the number of virtual warehouses, and how long they run to complete the activity.

- **Describe Snowflake Virtual Warehouse – Horizontal and Vertical Scaling?**

  Snowflake's virtual warehouses offer on-demand elastic scaling through a combination of vertical and horizontal scaling mechanisms. Vertical scaling (Scale up or down) allows for increasing the processing power of individual warehouses, ideal for handling complex queries. Horizontal scaling (Scale out) provides the ability to spin up additional warehouses, effectively distributing workloads across a larger pool of resources. This combined approach ensures optimal performance

for varying workloads while optimizing costs by paying only for the utilized resources.

- **Types of scaling policy available to Snowflake virtual Warehouse?**

  Snowflake offers two different scaling policies to handle the workload automatically, following are the details:

  - **STANDARD**: The default scaling policy is Standard, which prioritizes preventing query queuing over conserving credits.. It prevents or minimizes queuing of queries by favoring starting additional clusters over conserving credits.
  - **ECONOMY**: This policy conserves credits by starting additional clusters only when there are enough queries queued to justify the cost. This means that additional clusters will only be started when there are enough queries queued to justify the cost.

- **List the various types of data cache available in Snowflake?**

  Snowflake offers the following caching mechanism to optimize query performance, resource utilization and enhance the overall user experience to store the recent activities:

  - **Metadata cache**: This cache stores metadata about the data, such as the table schema and column types.
  - **Virtual warehouse cache**: This cache stores data that is frequently accessed by queries.
  - **Query result cache**: This cache stores the results of recently executed queries. When a user executes a query, Snowflake first checks the Query Result Cache to see if the results are already there. If they are, Snowflake returns the results from the cache, which is much faster than reading the data from the underlying storage.

- **List few best practices managing Snowflake virtual warehouses:**

  Following are few recommended best practices to manage the virtual warehouses:

  Following are few recommended best practices to manage the virtual warehouses:

  - **Monitoring warehouses:** User can monitor your warehouses to track their usage and performance. This information can help you to optimize your warehouse usage and to identify any potential problems.
  - **Adjust the auto-suspend time:** You can change the auto-suspend time based on the expected workload pattern.
  - **Changing the size of a warehouse:** You can change the size of a warehouse to increase or decrease the amount of compute resources that are available.

Chapter 9
Snowflake Performance Management

Introduction

This comprehensive chapter delves into the details of Snowflake's virtual warehouses, equipping you with the knowledge to make informed decisions for optimal performance, efficiency, and cost control.

We will cover the following key areas:

- **Fundamental understanding**: Explore various warehouse types, configurations, and scaling policies to tailor your environment to specific workloads.
- **Technical details**: Gain a deeper understanding of data caching mechanisms, warehouse management and monitoring techniques, and credit usage patterns.
- **Advanced strategies**: Master advanced features like auto-suspend/resume, fine-tuning parameters, and best practices for maximizing warehouse effectiveness.

Structure

In this chapter, we will cover the following topics:

- Performance overview
- Query profile
- Query performance analysis

- Query history and characteristics
- Optimizing query performance
- Materialized views and commands

Objectives

This chapter aims to provide comprehensive information about various performance characteristics of Snowflake including the Query profile, Query history, characteristics of Query history, how to analyze and improve the query performance using various commands and techniques, etc. By the end of this chapter, readers will gain knowledge to leverage Snowflake's warehouses like a seasoned professional, ensuring your queries run quickly, resources are utilized effectively, and your data warehousing endeavors remain cost-conscious.

Performance overview

Snowflake is a cloud-based data platform that offers high performance for a wide range of workloads. Snowflake built on a unique architecture that separates storage and computing, which allows it to scale independently. Snowflake also uses a number of other features to improve performance, such as **massively parallel processing** (**MPP**), columnar storage, and in-memory caching.

Snowflake's performance is measured by a number of key metrics, including query latency, throughput, and scalability. Snowflake typically achieves query latencies of milliseconds, even for complex queries and large datasets. It can process hundreds of millions of rows per second, making it well-suited for high-volume workloads. Snowflake can also scale to handle petabytes of data and thousands of concurrent users.

Snowflake's Performance Index (**SPI**) measures the real-world performance improvements that customers have experienced over time. According to Snowflake, the SPI has significantly increased over the past few years.

Snowflake is a high-performance cloud data platform that is well-suited for a wide range of workloads. There are a few things you can do to make Snowflake even faster:

- Choose the right warehouse size and type for your needs.
- Optimize your queries to run as efficiently as possible.
- Use materialized views to store pre-computed results of common queries.

Query profile

The Snowflake Query Profile is a powerful feature for understanding and optimizing the performance of your queries. It provides detailed insights into the execution of a query,

including a visual representation of the query plan, execution details and statistics for each node in the query plan, and query runtime stats.

The Query Profile can be used for a variety of purposes, such as:

- **Troubleshooting performance problems**: If a query is being performed slowly, the Query Profile can be used to identify the bottleneck. For example, the Query Profile might show that a particular operator is taking a long time to execute or that a large amount of data is being spilled to disk.
- **Identifying areas for query optimization**: The Query Profile can also be used to identify areas where a query can be optimized. For example, the Query Profile might show that a particular join is inefficient, or that an unnecessary sort is being performed.
- **Understanding the overall performance of your queries**: The Query Profile can also be used to understand the overall performance of your queries. For example, you can use the Query Profile to identify the queries that are taking the longest to execute or the queries that are consuming the most resources.
- **Comparing the performance of different versions of a query**: If you have made changes to a query, use the Query Profile to compare the performance of the different versions of the query. This can help to determine whether the changes you made have improved the performance of the query.
- **Learning more about how Snowflake executes queries**: The Query Profile can also be used to learn more about how Snowflake executes queries. This information can be helpful for understanding how to write more efficient queries.

To access the Query Profile for a query, navigate to Snowflake's **Query History** or **Worksheets** page in the Snowflake UI and click on the query ID. The detail page for the query will be displayed, and you can click on the **Query Profile** tab to view the Query details.

The Snowflake Query Profile is a robust diagnostic mechanism that empowers database administrators and developers to analyze and optimize query performance. By examining granular execution metrics, users can identify bottlenecks and inefficiencies that hinder longer query execution.

For example, prolonged query durations can be investigated through the Query Profile, potentially revealing a specific join operation as the primary cause. By modifying the join method to a more efficient alternative can significant reductions in overall query execution time.

In scenarios where a query consumes excessive memory resources and impacts concurrency, the Query Profile can ascertain if the query is frequently writing intermediate results to disk (spilling). Addressing this issue by leveraging existing materialized views can mitigate the spilling behavior, resulting in improved performance.

Moreover, when dealing with large datasets, the Query Profile can highlight if a query is scanning an excessive number of data partitions. Implementing appropriate strategies can streamline the query's data access patterns and lead to substantial performance gains.

In summary, the Snowflake Query Profile equips users with comprehensive insights into query execution, facilitating informed decision-making for query optimization and enhancing overall performance and better user experience.

Query performance analysis

Snowflake query performance analysis is the process of understanding and improving the performance of Snowflake queries. This is a critical process for optimizing the query efficiency. This entails identifying and resolving performance bottlenecks, refining query design, and ensuring optimal resource allocation.

Snowflake query performance analysis is important because it can help you to:

- **Enhanced performance**: By identifying and addressing performance bottlenecks, the overall performance of your Snowflake environment is significantly improved, resulting in cost reductions and increased user satisfaction.
- **Increased scalability**: Optimized query design and efficient resource allocation enable your Snowflake environment to scale effectively, accommodating larger workloads without compromising performance.
- **Proactive risk mitigation:** Regular analysis of Snowflake query performance allows for early detection of potential issues before they impact users, minimizing disruptions and ensuring a seamless experience.

There are a number of techniques that can be used to analyze Snowflake query performance, including:

- **Snowflake query profile**: Provides granular insights into query execution plans, step-by-step statistics, and overall runtime metrics. This enables in-depth analysis of individual queries, facilitating the identification of performance bottlenecks.
- **Snowflake performance management**: This integrated feature delivers real-time insights into account-level performance, encompassing resource utilization, query execution times, and historical query data.
- **Snowflake search optimization service**: Offers actionable recommendations to improve query performance. By leveraging this service, organizations can proactively **enhance** the efficiency and responsiveness of their Snowflake environment. This feature requires Snowflake Enterprise Edition (or higher).

To initiate the optimization process, prioritize the identification of queries that consume the most significant resources. Subsequently, utilize the Snowflake Query Profile to pinpoint specific performance bottlenecks and implement targeted optimizations. By

systematically addressing these issues, organizations can ensure optimal performance and resource utilization within their Snowflake environment.

Snowflake query performance can be impacted by several factors, including:

- **Suboptimal query design**: Inefficient join algorithms, delayed filtering, or excessive data retrieval can significantly hinder query performance.
- **Insufficient resource allocation**: Inadequate warehouse sizing and scaling policies can restrict the resources available to queries, resulting in slower execution times.
- **Data quality issues**: Incomplete or inaccurate data can slow down query processing, leading to performance degradation.

To address performance issues, it's crucial to identify the root cause. Once diagnosed, there are few potential solutions including query optimization, adjusting warehouse configurations to allocate more resources, and improving data quality etc.

Regularly analyzing Snowflake query performance is essential for proactively identifying and addressing bottlenecks before they impact users. This ensures optimal system performance and a positive user experience.

Recommendations for Analyzing Query Performance:

- **Leverage the Snowflake Query Profile**: Provides valuable insights into query performance bottlenecks.
- **Compare Query Versions**: Analyzing the performance of different query versions can reveal areas for optimization.
- **Monitor Performance Trends**: Tracking query performance over time helps identify long-term trends and potential issues.
- **Utilize the Search Optimization Service (Enterprise Edition or higher)**: This feature offers recommendations for enhancing query performance.
- **Consult with Snowflake Performance Experts**: When facing complex performance challenges, expert assistance can be invaluable.

By implementing these recommendations, organizations can optimize their Snowflake environment, ensuring efficient query execution and a superior user experience.

Query history and characteristics

Snowflake query history is a record of all the queries that have been executed on a Snowflake account. Snowflake query history is a valuable feature for troubleshooting performance issues, identifying slow-running queries, and monitoring user activity.

To monitor query activity within your Snowflake account, users can utilize the following options to view the query history:

- **Snowsight Query History Page**: This web-based interface provides a user-friendly way to visualize and analyze historical query data.
- **QUERY_HISTORY View (ACCOUNT_USAGE schema)**: This database view within the SNOWFLAKE database offers a structured format for accessing query history information using SQL queries.
- **QUERY_HISTORY Table Functions (INFORMATION_SCHEMA)**: This family of functions enables programmatic retrieval of query history data in various formats, allowing for further analysis and reporting.

Snowflake query history can be used for a variety of purposes, including:

- **Troubleshooting performance problems:** If a query is taking a long time to execute, you can check the query history to see what other queries are running at the same time. This can help you to identify the bottleneck.
- **Queued queries :** This can be identified using the query history if query is having more wait time than execution time - this is a warehouse issue.
- **Identifying trends and patterns:** You can use query history to identify trends and patterns in your query usage. For example, you can see which queries are taking the longest to execute, queued queries.
- **Auditing:** You can use query history to audit who is running what queries and when. This can help you to identify and unwanted or avoid any additional workload queries run as Adhoc.

Characteristics of Snowflake query history:

- **Comprehensive:** Snowflake query history includes information about all queries that have been executed on an account, regardless of the query type or the user who executed the query.
- **Query details**: Snowflake query history includes detailed information about each query, such as the query text, the query start time, the query end time, the query execution status, error info, and the resources consumed by the query and much more. This is also referred as metadata information of the query execution. This information can be used to identify long-running queries, queries that use a lot of resources, or queries that are failing.

Benefits of using Snowflake query history:

- **Improved query performance**: Snowflake query history can be used to track the performance of queries over time and to identify and troubleshoot performance bottlenecks. This can help improve query performance and cost reduction.
- **Increased scalability**: Snowflake query history can be used to identify queries that are consuming a lot of resources. This information can be used to optimize resource allocation and increase the scalability of Snowflake accounts.

Here are some examples of how the Query History can be used:

Example 1:

A query is taking a long time to execute, and you are not sure why. Review the Query History and that the query is spilling a large amount of data to disk. Optimize the query to use the appropriate summarized data or materialized view to improve the query performance.

Example 2:

You are migrating a large dataset from another data warehouse to Snowflake. Review the Query History to track the progress of the migration and to identify any performance bottlenecks that takes long time to execute. This query may be using an inefficient join algorithm. Optimize the query design and the query performance improves significantly.

The Snowflake Query History is a valuable feature for understanding, managing, and improving the performance of Snowflake. By regularly reviewing the Query History, you can identify and address performance bottlenecks, optimize query design, and ensure that your queries are performing optimally.

Additional things to know about Snowflake query history:

- Users can filter the query history by user, warehouse, query status, date range, also search the query history by keyword. This gives much flexibility to narrow down any level of details.
- Snowflake's query history retention varies depending on the access methods:
    - **Query history page**: This interface displays the most recent queries executed within the past 14 days.
    - **Information schema query history table function**: Similar to the Query History page, this function provides access to query history for the last 14 days.
    - **Account usage query history view**: This comprehensive view retains query history for a period of 365 days, allowing for detailed analysis across various dimensions such as time range, session, user, and warehouse.

Snowflake query history example

The **`QUERY_HISTORY`** table function is a powerful tool for monitoring and analyzing query execution activity in Snowflake. You can use it to filter and sort queries, generate reports on query execution performance, and identify queries that may be causing problems.

The following example code shows how to use the **`QUERY_HISTORY`** table function to get a list of all the queries that were executed in the last 2 days:

```
SQL:
-- Query History using the Information Schema Query History Table Function
select * from table(information_schema.query_history())
where DATE(START_TIME) >= CURRENT_DATE -2
order by start_time;
```

The above query will provide much useful information about the executed queries, following are few key columns:

- **QUERY_ID**: Unique identifier for the query.
- **QUERY_TEXT**: The text of the query.
- **USER_NAME**: User who executed the query.
- **WAREHOUSE_NAME**: The name of the warehouse where the query was executed.
- **START_TIME**: The time when the query started executing.
- **END_TIME**: The time when the query finished executing.
- **EXECUTION_STATUS**: The status of the query (for example, **SUCCESS**, **FAIL**, **CANCELLED**).

Users can use the Account Usage Query History View to create more complex queries that filter and sort queries by different criteria. For example, the following query returns a list of all the queries that were executed by the user john for the last 2 days:

```
SQL:
--Query History using the Account Usage Query History View
Select * from SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
where DATE(START_TIME) >= CURRENT_DATE -2
and USER_NAME in('JOHN')
order by START_TIME desc;
```

Optimizing Query performance

Snowflake query performance can be optimized in several techniques. To achieve optimal query performance in Snowflake, consider these key strategies:

- **Data access optimization:**
  - Minimize data scanning by selecting only required columns. Narrow data retrieval by explicitly selecting required columns instead of using wildcard (SELECT *).
  - Use pre-aggregated tables or materialized views for frequently used calculations to reduce query processing time.
  - Break down complex queries into manageable steps.

- **Join and data processing optimization:**
    - Utilize clustered columns in join predicates for improved join performance.
    - Leverage window functions over self-joins for aggregate calculations, fostering improved performance.
    - Avoid using OR conditions and complex views in joins.
    - Apply domain knowledge to help Snowflake efficiently process data by ordering data and using appropriate data types.
- **Performance enhancement techniques:**
    - Leverage the Snowflake Query Acceleration Service for automated optimization of query execution plans. This feature requires Snowflake Enterprise Edition (or higher).
- **Resource management and monitoring:**
    - Monitor query performance using Snowflake Query Details and Query Profile features.
    - Adjust virtual warehouse size and scaling policies to ensure sufficient resources for query execution by adjusting virtual warehouse configurations
    - **Monitor Disk Spillage**: Optimize data storage to minimize disk usage and enhance data access efficiency.

By implementing these best practices and utilizing Snowflake's comprehensive features, organizations can significantly enhance query performance and overall efficiency within the Snowflake data warehouse environment. It is crucial to experiment and adapt these strategies based on the specific characteristics of your data and workload to achieve optimal results.

Materialized views

Materialized views in Snowflake are pre-computed results of queries. They can be used to improve the performance of queries that use the same data repeatedly. Materialized views are especially useful for queries that return large amounts of data or that involve complex calculations. This feature requires Snowflake Enterprise Edition (or higher).

Materialized views in Snowflake can significantly enhance query performance by pre-computing and storing query results. However, the act of materializing and storing these intermediate results incurs storage and computational overheads. These views consume additional storage resources to store results and require compute resources for automatic background maintenance to keep them synchronized with the base tables. Snowflake charges for actual compute resources used billed in one-second increments. It is advisable to carefully evaluate the trade-off between the performance gains and the associated costs before implementing materialized views in Snowflake.

To create a materialized view in Snowflake, you use the **`CREATE MATERIALIZED VIEW`** statement. The following example shows how to create a materialized view that contains the total sales for each region:

```
SQL :
-- Sample Syntax/Code to Create MATERIALIZED VIEW
CREATE OR REPLACE MATERIALIZED VIEW total_sales_by_region
AS
SELECT region, SUM(sales) AS total_sales
FROM sales_table
GROUP BY region;
-- Sample Syntax/Code to query the MATERIALIZED VIEW
Select * from total_sales_by_region;
```

Materialized views are recommended when query results are relatively small compared to the base table, involve computationally intensive processes, or are derived from slower external data sources. They are particularly advantageous when the underlying data is infrequently updated. Materialized views can be used to improve the performance of a wide variety of queries, including:

- **Reporting queries**: Materialized views can be used to improve the performance of reporting queries that return large amounts of data or that involve complex calculations.
- **Data warehouse queries**: Materialized views can be used to improve the performance of data warehouse queries that are used to analyze large datasets.
- **Real-time analytics queries**: Materialized views can be used to improve the performance of real-time analytics queries that need to return results quickly.

Deciding when to use materialized view or regular view

When deciding between a materialized view and a regular view, consider the following factor that fit your organization need:

- **Frequency of data changes**: If the underlying data changes infrequently, a materialized view is preferred as it stores the query results, reducing the need for frequent recalculation. However, if the data changes frequently, a regular view might be more suitable to ensure the latest data is reflected.
- **Frequency of view usage**: For frequently accessed queries, a materialized view can significantly improve performance by providing pre-computed results. Conversely, if the view is seldom used, the additional storage cost of a materialized view may not justify the performance gain.

- **Resource intensity of query**: Complex and resource-intensive queries can benefit from materialized views, as they offload the processing burden. However, for simple queries, the performance improvement might be negligible compared to the overhead of maintaining the materialized view.
- **Additional considerations**: Beyond these factors, storage costs and the specific use case should be evaluated. While a materialized view might seem appealing for performance, it might not be cost-effective if the query results are rarely used. It's crucial to assess your specific use case and weigh these factors to determine the optimal view type for your requirements.

Ultimately, a thorough analysis of the query characteristics, data volatility, and performance requirements is crucial for determining whether a materialized view or a regular view is the most appropriate solution for a specific scenario.

Conclusion

In this chapter, we learned about various performance characteristics of Snowflake that includes the Query profile, Query History and characteristics of Query History, how to analyze Snowflake query performance etc. We also learned about the second part of the performance topics and details on how to improve the query performance using various commands and techniques like Materialized views.

In the next chapter, we will learn about how the data gets loaded and unloaded to and from Snowflake, and additional technical details around the Snowflake data load/unload process, such as stages, file formats, file size, and so on.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **How to access Snowflake Query Profile and list few Benefits of using it?**

  The Snowflake Query Profile is a powerful feature for understanding and optimizing the performance of your queries. To access the Query Profile, Navigate to your Query History or Worksheets in Snowflake UI and click on the query you want to look at. This provides detailed insights into the execution of a query, including a visual representation of the query plan, execution details and statistics for each node in the query plan, and query runtime stats.

  Key benefits of utilizing the Query Profile include:

  - **Performance troubleshooting**: Identify bottlenecks and inefficiencies in query execution, such as slow-performing operators or excessive disk usage.
  - **Query optimization**: Pinpoint areas for improvement, such as inefficient joins or unnecessary sorting operations.

  - **Performance monitoring**: Gain insights into overall query performance, identifying long-running or resource-intensive queries.
  - **Comparison analysis**: Evaluate the impact of query modifications by comparing the performance of different versions.
  - **Educational resource**: Enhance understanding of Snowflake's query execution engine, leading to the development of more efficient queries.

- **Why should organizations care about query performance Analysis?**

  Snowflake Query Performance Analysis (QPA) is the process of checking and improving how fast and well Snowflake queries work. This involves finding and fixing issues that slow things down, making queries better, and using resources wisely.

  QPA is important because it helps:

  - **Make things faster**: By fixing problems, Snowflake can run more smoothly and quickly.
  - **Handle more work**: By improving queries and managing resources, Snowflake can handle larger tasks without slowing down.
  - **Avoid problems**: By checking regularly, potential issues can be found and fixed before they cause trouble.

  By regularly checking and fixing queries, you can keep Snowflake running at its best.

- **What is Snowflake Query History and how to access it?**

  Snowflake query history is a record of all the queries that have been executed on a Snowflake account. Snowflake query history is a valuable feature for troubleshooting performance issues, identifying slow-running queries, and monitoring user activity.

  To monitor query activity within your Snowflake account, users can utilize the following options to view the query history:

  - Snowsight Query History Page
  - **`QUERY_HISTORY`** View (**`ACCOUNT_USAGE`** schema)
  - **`QUERY_HISTORY`** Table Functions (**`INFORMATION_SCHEMA`**)

- **What is the query History retention for Snowflake using varies options?**

  Snowflake offers varying query history retention periods depending on the method of access:

  - **Query History Page and Information Schema Query History Table Function**: These interfaces provide access to the most recent 14 days of query history.
  - **Account Usage Query History View**: This comprehensive view offers a robust retention period of 365 days, facilitating in-depth analysis across multiple dimensions including time range, session, user, and warehouse utilization.

- **What is a Materialized View?**

  Materialized views within Snowflake offer significant query performance enhancements by pre-computing and storing query results. This is particularly beneficial for complex or frequently executed queries involving large datasets. Required Snowflake Enterprise Edition (or higher) subscription.

- **List a few benefits of using the materialized Views:**

  Following are a few key benefits of using Snowflake's Materialized views:

  - **Enhanced query performance**: Materialized views pre-compute and store aggregated results, enabling significantly faster query response times for complex or frequently executed queries.
  - **Optimized resource utilization**: By reducing the need for on-the-fly calculations, materialized views minimize computational overhead and optimize resource allocation within Snowflake.
  - **Improved data freshness**: Automatic refresh mechanisms ensure that materialized views remain synchronized with underlying data sources, guaranteeing access to up-to-date information.
  - **Simplified data access**: Materialized views present a streamlined interface for accessing complex or aggregated data, promoting efficient data consumption for end-users and applications.
  - **Cost efficiency**: The reduced computational demands of materialized views can translate to cost savings within Snowflake's consumption-based pricing model.

- **When to use Materialized view or Regular View?**

  When deciding between a materialized view and a regular view, consider the following factors that fit your organization need:

  Use a materialized view if:

  - The data in the view rarely changes. This means the underlying tables are mostly static.

- The view is used frequently. It's worth the extra storage space if the view gets a lot of traffic.
- The query is complex. A materialized view can save time and resources if the query is computationally heavy.

Use a regular view if:

- The data in the view changes often. A regular view is updated in real-time, reflecting the latest data.
- The view is used infrequently. It's not worth the extra storage space if the view is rarely used.
- The query is simple. A regular view is sufficient if the query is straightforward and doesn't require a lot of processing.

Remember: Materialized views take up additional storage resources to store results and require compute resources for automatic background maintenance to keep them synchronized with the base tables. So consider if the speed improvement is worth the extra cost. It is advisable to carefully evaluate the trade-off between the performance gains and the associated costs before implementing materialized views in Snowflake.

Chapter 10
Snowflake Data Loading and Unloading

Introduction

This chapter delves into the essential aspects of data movement within Snowflake. We will explore the technical intricacies involved in loading and unloading data, including:

- **Stages**: Discover how Snowflake utilizes stages as temporary storage locations for data files during the load/unload process.
- **File formats**: Understand the supported file formats (for example, CSV, Parquet) and their significance for efficient data handling. Learn strategies for managing file sizes to optimize performance and resource utilization.
- **Data loading and unloading methods**: Explore various techniques for loading data into Snowflake and unloading data from it, catering to diverse needs and scenarios.

Structure

In this chapter, we will cover the following topics:

- Snowflake stages
- File size and formats
- Data loading methods

- Data unloading methods
- Concepts and best practices to load data
- Concepts and best practices to unload data

Objectives

This chapter aims to provide comprehensive information about Snowflake data movement, and other technical components and fundamental concepts like stages, file formats, file size considerations, and various methods and commands involved in this data movement process. By the end of this chapter, readers will be equipped with the knowledge and skills to confidently manage data movement within your Snowflake environment.

Snowflake stages

A Snowflake stage is a temporary storage location of data files within cloud storage as a *stage*. The **`COPY INTO <table>`** command, utilized for both bulk loading and continuous data ingestion (Snowpipe), accommodates cloud storage accounts managed by your organization (external stages) as well as storage residing within your Snowflake account (internal stages). This flexibility allows seamless data loading from diverse sources, catering to your specific infrastructure and data management preferences.

Snowflake supports loading data from Amazon S3, Google Cloud Storage, and Microsoft Azure, regardless of the cloud platform hosting your Snowflake account. However, data stored in archival storage classes requiring restoration (for example, Amazon S3 Glacier Flexible Retrieval or Glacier Deep Archive, Microsoft Azure Archive Storage) is not directly accessible. We will learn more information in the next section that explains different stage types supported by Snowflake and further technical details.

External stages

External stages are used to store data in cloud storage buckets. Files can be uploaded to your cloud storage using the service's native tools. Snowflake utilizes named external stages (database objects defining file URLs, access credentials, and format options) to interact with this data. External stages are created using the **`CREATE STAGE`** command and must specify the location of the data files. External stages can be created in any schema and can be accessed by any user who has the appropriate permissions. Snowflake external stages can be configured to following cloud storage providers:

- Amazon S3
- Microsoft Azure Blob Storage
- Google Cloud

These storage locations can be either public or private or protected. Prior configuring the external stages that are private or protected, user required to add the storage integration

to the external cloud storage. This requires secrete access key or credentials to setup secure connection to the cloud storage bucket on any cloud.

- **Storage integration**: Snowflake strongly recommends leveraging storage integrations for configuring external stages. This strategic approach enables Snowflake to seamlessly access and process data stored in these external locations eliminating the need for input of cloud storage credentials during the creation of stages or data loading processes. Storage integration in Snowflake serves as a secure and efficient gateway between the Snowflake data warehouse and external cloud storage providers such as Amazon S3, Google Cloud Storage, or Azure. To establish the integration, execute the **`CREATE STORAGE INTEGRATION`** command, providing the necessary parameters for your cloud storage provider.
- **Understand data transfer cost**: Transferring data into a Snowflake account incurs no charge. However, your cloud storage provider may charge egress fees for moving data from their platform to Snowflake. Snowflake data transfer cost based on the direction and destination of the data movement scenarios. Following are the high level details:
    - **Data Ingress (Moving data into Snowflake)**: Snowflake does not directly charge for data ingress. However, your cloud storage provider (for example, Amazon S3, Google Cloud Storage, or Microsoft Azure) may have data egress fees when transferring data from their platform to your Snowflake account. Please consult your provider for specific details on potential egress charges.
    - **Data Egress (Moving data out of Snowflake)**: Snowflake incurs a per-byte fee for data egress when transferring data to a different region on the same cloud platform or a different cloud platform altogether. Data transfers within the same cloud region are free and no additional cost involved. The per-byte rate varies depending on your Snowflake account's hosting region. These charges are applicable to functionalities that involve the egress of data from your Snowflake account. Following are few scenarios where data transfer costs may arise include:
        - **Data unloading**: Unloading data from Snowflake to external cloud storage services like Amazon S3, Google Cloud Storage, or Azure Blob Storage.
        - **Data replication**: Establishing a secondary copy of your database in a geographically distinct region or an alternative cloud environment.
        - **External network access**: Initiating connections to external resources from within Snowflake's stored procedures or user-defined functions (UDFs).

- **External function utilization**: Utilizing functions designed to facilitate data transfer to major cloud providers such as AWS, Microsoft Azure, or Google Cloud Platform.
- **Cross-cloud auto-fulfillment**: Facilitating the automatic provisioning of listings to consumers in diverse cloud regions.

Please note that Snowflake does not levy egress charges for data retrieval operations conducted across regions within the same cloud platform or across different cloud platforms when using a Snowflake client or driver.

Internal stages

Internal stages are used to store data files in Snowflake's own storage layer. Internal stages can be used to store data temporarily before it is loaded into a table or to store data that is being unloaded from a table. Internal stages are created using the **`CREATE STAGE`** command and do not need to specify the location of the data files, as they are already stored in Snowflake. Internal stages can be created in any schema and can be accessed by any user who has the appropriate permissions.

Snowflake supports the following three distinct types of internal stages to facilitate efficient data management within your account. Regardless of the stage type, file uploads from local file systems can be seamlessly executed using the **`PUT`** command.

- **User stages:** Each user is allocated a dedicated stage for individual file storage. Designed for individual use, these stages facilitate the loading of data into multiple tables. User stages are not modifiable or removable.
- **Table stages:** Automatically provisioned for each table, these stages are designed to store files intended for loading into a specific table, but accessible by multiple users. Ownership privileges on the table are required for managing files within a table stage. These stages cannot be altered or deleted.
- **Named stages:** As database objects, named stages offer the most versatile solution for storing files. They can be accessed by multiple users, loaded into multiple tables. They can be managed using standard access control privileges and created using the **`CREATE STAGE`** command.

Refer to the following section for the sample syntax/code for internal external stages:

```
SQL :
-- Sample Syntax/Code to Create Internal stage
CREATE STAGE my_sample_internal_stage
  ENCRYPTION = (TYPE = 'SNOWFLAKE_SSE');  -- Server-Side Encryption
(recommended)
-- Optionally, enable the directory table for file listings:
ALTER STAGE my_sample_internal_stage
```

```
  SET DIRECTORY = (ENABLE = TRUE);
```

Refer to the following section for the sample syntax/code for creating external stages:

```
SQL :
-- Sample Syntax/Code to create storage integration for private/protected
AWS S3 bucket
CREATE  STORAGE INTEGRATION mys3_ext_int
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'S3'
  STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::1011121315:role/myrole'
  ENABLED = TRUE
  STORAGE_ALLOWED_LOCATIONS = ('*')
  STORAGE_BLOCKED_LOCATIONS = ('s3://testbucket/path1/', 's3://testbucket2/
path2/');
-- Sample Syntax/Code to create external Stage using storage integration
CREATE STAGE mys3_ext_stage
  COMMENT = 'External stage using Storage Integration for private/protected
AWS S3 bucket'
  URL='s3://load/files/'
  STORAGE_INTEGRATION = mys3_ext_int;
-- Sample Syntax/Code to create external Stage for private/protected AWS S3
bucket using IAM credentials
CREATE STAGE mys3_ext_stage2
  COMMENT = 'External stage using IAM credentials for private/protected AWS
S3 bucket'
  URL='s3://test2/files/'
  CREDENTIALS=(AWS_KEY_ID='xygdfs' AWS_SECRET_KEY='we5tgs');
-- Create an external Stage pointing to AWS S3 public bucket
CREATE STAGE mys3_ext_stage3
  COMMENT = 'External stage - S3 public bucket'
    url = 's3://test3/Sample_files/';
```

Once a stage has been created, it can be used to load data into Snowflake tables using the **`COPY INTO`** command. We can further learn the details and steps in the upcoming section.

Here are additional considerations when working with the stages:

- Note that **`CREATE STAGE`** does not validate the specified URL or credentials. Invalid credentials will result in an error upon stage utilization.
- Recreating a stage within a database environment (utilizing the **`CREATE OR REPLACE STAGE`** command) can result in unintended consequences that require further action:

- **Loss of directory association**: The existing directory table linked to the stage, if present, will be dropped. If a new directory is created, it will be initialized as empty.
- **Disruption of external table links**: Any external tables referencing the stage will experience a broken association. This occurs due to the internal mechanism of external table linkage, which relies on a hidden identifier rather than the stage name. The `CREATE OR REPLACE` operation inherently generates a new identifier, thus severing the connection.
- **Interruption of data pipelines**: Any pipes configured to utilize the stage for data loading will cease operation. The execution status of these pipes will transition to `STOPPED_STAGE_DROPPED`.

Carefully consider the potential consequence prior to executing a `CREATE` or `REPLACE STAGE` operation to minimize disruptions to dependent database objects and processes.

Snowflake stages act as a streamlined interface between external data sources and Snowflake's processing capabilities. By staging data, organizations can easily ingest, transform, and analyze data from diverse sources within a unified platform. This enhances data accessibility, improves operational efficiency, and empowers organizations to derive valuable insights from their data assets.

Snowflake file format

In Snowflake, a file format is a reusable object that encapsulates the metadata defining the structure and layout of data files. This metadata includes attributes such as file type, compression method, field and record delimiters, row separator, and other format-specific options. By defining a file format, data engineers can streamline the process of loading data into or unloading data from tables in Snowflake, ensuring consistent and accurate data ingestion and extraction, and thereby optimizing data pipeline efficiency.

Snowflake file formats used to specify the data types of the columns in the data file. This can help to improve the performance of the data loading process and can also help to ensure that the data is loaded correctly into the Snowflake table.

Snowflake supports a variety of file formats for both structured and semi-structured data, including:

- **Delimited files (for example, CSV, TSV)**: Simple text files where values are separated by commas or tabs. Great for basic, tabular data.
- **JSON**: A flexible format for semi-structured data that can be easily read by humans.
- **Avro, Parquet, and ORC**: Columnar storage formats designed for efficient processing of large datasets.
- **XML**: A markup language used to represent hierarchical data structures.

The choice of file format depends on various factors, including data structure, volume, and intended use cases. Organizations can leverage Snowflake's extensive format support to build robust and scalable data solutions that cater to their unique requirements. Following are few commonly used characteristics of the file formats, but there are many configurations available based on file type selection and your organization need:

- **Field delimiter**: The character that separates fields in the data file.
- **Field enclosure**: The character that encloses fields in the data file.
- **Null value**: The value that represents a null value in the data file.
- **Escape character**: The character that is used to escape special characters in the data file.
- **Date and time formats**: You can customize how dates and times are interpreted in the data files.
- **Character set**: You can specify the character set of the data files (for example, UTF-8, ISO-8859-1).

To use a Snowflake file format, you must first create the file format object using the **`CREATE FILE FORMAT`** statement. Once you have created the file format, you can use it to load or unload data from Snowflake tables.

Refer to the following section for the sample syntax/code for creating various file formats:

```
SQL:
-- Sample Syntax/Code to Create File Format
create file format <name>
    type = { CSV | JSON | AVRO | ORC | PARQUET | XML }
    --  [ formatTypeOptions ]
    -- comment = '<comment>'
-- Sample Syntax/Code to Create CSV File Format
CREATE FILE FORMAT csv_file_format
  TYPE = CSV
  FIELD_DELIMITER = ','
  SKIP_HEADER = 1
  COMPRESSION = gzip;
-- Sample Syntax/Code to Create JSON File Format
CREATE FILE FORMAT json_file_format
  TYPE = JSON;
-- Sample Syntax/Code to Create AVRO File Format
CREATE FILE FORMAT avro_file_format
TYPE = AVRO;
-- Sample Syntax/Code to Create PARQUET File Format
```

```
CREATE FILE FORMAT parquet_file_format
  TYPE = PARQUET
  COMPRESSION = SNAPPY;
-- Sample Syntax/Code to Create XML File Format
CREATE FILE FORMAT xml_file_format
TYPE = XML;
```

Here are additional considerations when working with the file formats:

In Snowflake, recreating a file format using **`CREATE OR REPLACE FILE FORMAT`** invalidates its association with any external tables referencing it. This occurs because external tables link to file formats via an internal identifier, distinct from the user-defined name, which is altered upon file format replacement. To rectify this issue, it is imperative to recreate any external tables utilizing the original file format. The **`GET_DDL`** function within Snowflake can be leveraged to retrieve the necessary DDL statement for each table's recreation.

Once you have created the file format object, these file formats can be used to load or unload data in Snowflake. Here are some examples of how Snowflake file formats can be used:

- **Loading data from a cloud storage bucket**: You can use a Snowflake file format to load data from a cloud storage bucket, such as Amazon S3, Microsoft Azure Blob Storage, or Google Cloud Storage. This can be useful for migrating data to Snowflake, or for loading data from external sources into Snowflake tables for analysis.
- **Unloading data to a cloud storage bucket**: You can also use a Snowflake file format to unload data from Snowflake to a cloud storage bucket. This can be useful for backing up data, or for exporting data from Snowflake tables to other systems for further analysis or processing.

Snowflake file formats can be used to load and unload data in a variety of ways. By understanding Snowflake file formats, you can improve the performance and reliability of your data pipelines. Refer to the following table for high-level comparisons for various file formats and use cases supported by Snowflake:

File Format	Data Structure	Loading Support	Unloading Support	Primary Use Cases
CSV	Structured	✓	✓	Supports any delimiter; highly versatile for various data types.
JSON	Semi-structured	✓	✓ (NDJSON)	Supports standard JSON and NDJSON (newline delimited JSON) formats.
Parquet	Semi-structured	✓	✓	Columnar format for fast data analysis.

File Format	Data Structure	Loading Support	Unloading Support	Primary Use Cases
Avro	Semi-structured	✓	X	Supports efficient storage and processing of large datasets.
ORC	Semi-structured	✓	X	Optimized columnar format for analytical workloads.
XML	Semi-structured	✓	X	Supports hierarchical data structures; often used for configuration files.

Table 10.1: *Snowflake File format matrix*

Snowflake file size

Snowflake supports files of any size, but it is recommended to keep file sizes between 100 MB and 250 MB for optimal performance. This is because Snowflake loads data in parallel and having a larger number of smaller files allows Snowflake to distribute the load more evenly. Few reasons behind this recommendation are as follows:

- **Parallel processing**: Snowflake can load multiple files in parallel, which can improve the overall performance of the load process. Using smaller files allows Snowflake to start loading more files in parallel, which can lead to faster load times.
- **Storage efficiency**: Snowflake stores data in compressed format and uses micro partitions to create data chunks that are used for query processing using pruning mechanism, which can store the data more efficiently than other formats.
- **Performance**: Smaller file sizes can improve performance, as they can be loaded into Snowflake more quickly. However, too many small files can also impact performance, as Snowflake needs to process each file individually.
- **Efficiency**: Larger file sizes can be more efficient, as they require fewer resources to load into Snowflake. However, too large of a file can impact performance, as it can take longer to load into Snowflake.
- **Error handling**: If you are loading a large number of files into Snowflake, it is important to consider how you will handle errors. If a file fails to load, Snowflake will stop the entire load process. To avoid this, you can use the ON_ERROR clause to specify how Snowflake should handle errors.

Here are some best practices for Snowflake file size:

- Use file sizes that are 100-250 MB in size (compressed).
- If you need to load a very large file, consider splitting it into smaller files before loading. You can use a tool such as split to split the file into smaller chunks.

- **Test different file sizes**: The best way to determine the ideal file size for your Snowflake data loads is to experiment. Try loading your data using different file sizes and see which file size performs the best.
- **Use the `ON_ERROR` clause**: If you are loading a large number of files into Snowflake, it is important to use the **`ON_ERROR`** clause to specify how Snowflake should handle errors. This will help to ensure that the entire load process does not fail if a single file fails to load.

By following these best practices, you can choose the right file size for your Snowflake data loads and improve the performance and efficiency of your data pipelines.

Here are some additional tips for loading data into Snowflake:

- Use a compressed file format. Compressing your files before loading them into Snowflake is highly recommended, as it reduces storage costs and can improve loading performance.
- Use a file format that is supported by Snowflake. Snowflake supports a variety of file formats, including CSV, JSON, Avro, ORC, Parquet, and XML.
- Use a consistent file format for all of the files in a load operation. This will help to improve performance and avoid loading errors.
- Parallelism: Snowflake can load multiple files in parallel, so breaking large files into smaller chunks can often improve performance.

Data loading methods

Snowflake offers multiple data loading options. This overview outlines the fundamental concepts, tasks, tools, and techniques to efficiently and seamlessly integrate data into your Snowflake database.

Supported file locations: Snowflake utilizes the term *stage* to denote the location of data files within cloud storage. The **`COPY INTO <table>`** command, used for both bulk and continuous data loading (via Snowpipe), supports: external stages (cloud storage accounts managed by your organization) and internal stages (cloud storage within your Snowflake account).

Snowflake offers comprehensive solutions for data loading, catering to diverse needs and data volumes. The optimal approach depends on specific requirements, such as data size and frequency of updates. Following are the two types of data load methods:

- **Bulk loading:** This method leverages the **`COPY`** command to efficiently load large batches of data from files residing in cloud storage or local machine. It supports simple transformations like column reordering, omission, casting, and truncating text string that surpass the target table column, and so on, during the loading process, streamlining data preparation.

- **Continuous loading**: This process utilizes the continuous load using Snowpipe. Designed for real-time data ingestion, this approach seamlessly integrates with various data sources, ensuring Snowflake tables are consistently up-to-date. It is particularly well-suited for scenarios where data is generated continuously.
- **Key consideration**: Bulk loading necessitates a user-provisioned virtual warehouse to manage the compute resources required for optimal performance. It is crucial to size the warehouse appropriately to accommodate the anticipated data volume.

Choosing the right loading method is paramount. For large, infrequent loads, bulk loading offers efficiency. However, continuous loading provides a superior solution for ongoing data streams, ensuring data freshness and minimizing latency. By understanding the unique capabilities of each approach, organizations can make informed decisions that align with their data loading requirements and overall data management strategy.

Bulk loading

To load data from a file using the **`COPY INTO`** statement, you must first create a Snowflake stage and a file format. We learned this in the prior section how to create Stage and file formats in Snowflake. Once you have created a stage and file format, you can use the **`COPY INTO`** statement to load the data from the stage into a Snowflake table. The **`COPY INTO`** statement will parse the data files and load the data into the table.

Here is an example of how to load data from a CSV file in a cloud storage bucket into a Snowflake Sales table. This code will create a Snowflake Sales table and load data using previously created stage.

```
SQL :
-- Sample Syntax/Code to create a table for data load
CREATE OR REPLACE TABLE sales(
      DATE DATE,
    CITY VARCHAR(16777216),
    ITEM_NAME VARCHAR(16777216),
    TOTAL_SALES NUMBER(17,0)
);
-- Ingest data using the previously created Stage (This is an example,
change code appropriately)
COPY INTO sales
    FROM @mys3_ext_stage3/salesfile.csv;
```

Continuous loading using Snowpipe

Snowpipe is a feature within Snowflake designed for continuous, near-real-time data ingestion. It efficiently loads micro-batches of data, making them available for analysis within minutes of arrival.

Leveraging a serverless compute model, Snowpipe dynamically scales resources based on demand, ensuring optimal performance and cost-efficiency through per-second billing. This means you only pay for what you actually consumed during data loading, with no need to provision or manage virtual warehouses. Snowflake charges calculated based on a per-second, per-core basis, and the usage is translated into Snowflake credits.

Basic transformations can be applied during the loading process, mirroring the capabilities of bulk loading operations. Furthermore, Snowpipe integrates seamlessly with data pipelines for managing complex transformations. Raw data is loaded into a staging area, then automated tasks and streams within the pipeline progressively refine and optimize the data for analysis. This architecture is particularly well-suited for scenarios involving continuous data streams.

```
SQL :
-- Sample Syntax/Code to create a table for Snowpipe data load
CREATE OR REPLACE TABLE my_stage_table(
  DATA VARIANT );
-- Sample Syntax/Code to create Snowpipe for continuous data load (This is
an example, change code appropriately)
CREATE PIPE my_snowpipe
  AS
  COPY INTO my_stage_table
  FROM @mys3_ext_stage
  FILE_FORMAT = (TYPE = 'JSON');
```

Key consideration when recreating Snowpipe for automated data loads:

When modifying a Snowpipe (using **`CREATE OR REPLACE PIPE SQL`**) configuration, especially one automating data loads, it's necessary to recreate the pipe to implement the changes. This process requires careful execution to avoid data duplication or loss of automated loading functionality.

Here are the recommended steps for recreating a Snowpipe:

- Pause the Snowpipe: Execute an **`ALTER PIPE ... SET PIPE_EXECUTION_PAUSED =`** true statement to temporarily halt the pipe's operation.
- Verify Pipe Status: Query the **`SYSTEM$PIPE_STATUS`** function to ensure the pipe's execution state is **`PAUSED`**.
- Recreate the Pipe: Utilize the **`CREATE OR REPLACE PIPE`** statement to generate a new pipe with the desired modifications.
- Pause the New Pipe: Temporarily pause the newly created pipe using the same **`ALTER PIPE`** statement.

- **Review cloud configuration**: Ensure that your cloud messaging service (for example, Amazon S3, Google Cloud Storage, Azure Blob Storage) configuration remains accurate and aligned with the updated pipe.
- **Resume the new pipe**: Set `PIPE_EXECUTION_PAUSED = false` via the `ALTER PIPE` statement to reactivate the pipe.
- **Verify pipe status**: Query the `SYSTEM$PIPE_STATUS` function to confirm that the new pipe's execution state is `RUNNING`.

Snowpipe maintains a load history within the pipe object's metadata. If you recreate the pipe, this load history gets removed. This usually does not cause problems unless you later use a command called `ALTER PIPE...REFRESH`. This command could accidentally load the same data twice from files that are still waiting to be processed if that data was already successfully loaded before and those files were not deleted.

Continuous loading using Snowpipe streaming

Snowpipe Streaming is a Snowflake feature that enables the continuous loading of data directly into tables without the need for staging files. This architecture results in reduced load latency and lower costs, making it a valuable tool for handling real-time data streams of any volume.

Snowpipe Streaming is also available for the Snowflake Connector for Kafka, offering a straightforward upgrade path to leverage the benefits of faster and more cost-effective loading. By utilizing the Snowpipe Streaming API in conjunction with the Snowflake Ingest SDK and your own managed application code, you can achieve low-latency ingestion of streaming data rows.

This approach fundamentally differs from bulk loads or traditional Snowpipe, which rely on staging files. Snowpipe Streaming writes data rows directly to tables, resulting in improved performance and cost efficiency, particularly for real-time data scenarios.

Snowpipe streaming vs. Snowpipe

Snowpipe Streaming API is a new offering designed to complement Snowpipe, not replace it. It is ideal for streaming data scenarios where data is streamed via rows (for example, Apache Kafka topics) rather than written to files. The API seamlessly integrates into existing ingestion workflows that utilize a custom Java application for producing or receiving records. By eliminating the need to create files for data loading into Snowflake tables, the API enables automatic and continuous loading of data streams as soon as they become available. This results in a more efficient and streamlined data ingestion process for organizations with streaming data requirements.

Refer to the following table for high-level comparisons between Snowflake streaming and Snowpipe data loading methods:

Category	Snowpipe Streaming	Snowpipe
Form of data to load	Rows	Files (If your existing pipeline generates files, Snowpipe is recommended)
3rd-party software	Custom Java application with the Snowflake Ingest SDK	None
Data ordering	Ordered insertions within each channel	Not supported (file load order may differ from file creation time in the cloud storage)
Load history	Recorded in **`SNOWPIPE_STREAMING_FILE_MIGRATION_HISTORY`** (Account Usage) view	Recorded in **`LOAD_HISTORY`** view (Account Usage) and **`COPY_HISTORY`** function (Information Schema)
Pipe object	Not required (API writes directly to tables)	Required (queues and loads data into tables)

Table 10.2: *Snowpipe streaming vs. Snowpipe loading*

Deciding which data loading method you choose will depend on your specific needs. If you need to load a large amount of data into Snowflake quickly, then direct loading is the best option. If you need to load data into Snowflake in real time or near-real time, then Snowpipe is the best option.

Concepts and best practices to load data

This section provides best practices, guidelines, and considerations for efficiently loading large volumes of data into Snowflake tables using the **`COPY INTO <table>`** command. It is intended to streamline and optimize the process of importing data from files into your Snowflake environment. Here are some best practices for loading data into Snowflake:

- **Preparing data files:** To ensure optimal performance and resource utilization when loading data into Snowflake, adhere to the following file sizing recommendations for both bulk loads and continuous loading using Snowpipe:
    - **Target file size**: 100-250 MB compressed: This size range strikes an effective balance between maximizing parallel processing and minimizing overhead.
    - **File size considerations**: Avoid very large files (>100 GB): These can cause processing delays, potential errors, and inefficient resource consumption.
    - **Split large files**: Partition into smaller files within the recommended range (100-250 MB).
    - **Error handling**: Implement appropriate error handling strategies to prevent data loss when dealing with large files.

  - **Recommended file size for semi-structured data**: Snowflake provides robust support for semi-structured data, such as JSON, with a maximum data size of 16 MB per item. For large JSON files exceeding 16 MB, Snowflake's **`STRIP_OUTER_ARRAY`** file format option enables efficient loading by parsing individual records into separate table rows.
  - **Continuous Data Loading (Snowpipe)**: For optimal Snowpipe performance and cost-efficiency, data files should ideally range from 100-250 MB or larger.

- **Dedicate resources for various workloads**: To maintain optimal performance when working with large datasets, it is advisable to implement separate warehouses for data loading and query execution. This separation ensures dedicated resources for each process, mitigating potential resource contention. The warehouse's compute resources directly determine the volume of data that can be processed concurrently.
- **Stage your data before load:** When staging data sets, a strategic partitioning approach is recommended. By segmenting data into logical paths that incorporate identifiers such as geographical location, source origin, and data write date, organizations can enhance data retrieval efficiency and granularly control data loading processes. This structured methodology facilitates the selective importation of specific data subsets and optimizes query execution, ultimately improving overall system performance.
- **Manage regular data loads:** Best practice to manage the regular data load by organizing your data by creating a clear folder structure, specifying the exact location of your data and cleaning up the original files after a successful load.

By adhering to these best practices, you can streamline your data loading workflows, optimize resource utilization, and ensure efficient data ingestion into your Snowflake environment.

Data unloading methods

Similar to data loading, Snowflake supports bulk export to unloading of data from database tables into flat, delimited text files. Snowflake's data unloading process mirrors its loading process, but in reverse. Data unloading involved the following two step process.

- **Export data**: Utilize the **`COPY INTO <location>`** command to transfer data from your Snowflake table into one or more delimited text files. These files can reside within a Snowflake internal stage or an external stage like AWS S3 or Azure Blob Storage or GCP.
- **Download the files from Snowflake internal/external stage**:
  - **Snowflake Stage**: For files within a Snowflake stage, utilize the **`GET`** command to download them to your local environment.

- **Amazon S3**: Leverage Amazon's suite of tools and APIs (for example, AWS CLI, S3 Console) to securely download your files.
- **Azure Blob Storage**: Utilize Microsoft Azure's tools and APIs (e.g., Azure Storage Explorer) to download your data files.

This process facilitates efficient and secure extraction of data from your Snowflake environment, allowing integrating it with other systems or performing further analysis and processing.

SQL :

```
-- Sample Syntax/Code to unload sales table data to an external stage (AWS
s3) utilizing the storage integration (This is an example, change code
appropriately)
COPY INTO 's3://myextbucket/dataunload/'
  FROM sales
  STORAGE_INTEGRATION = mys3_ext_int
  FILE_FORMAT = (FORMAT_NAME = csv_file_format);
```

Concepts and best practices to unload data

This section provides best practices, guidelines, and considerations for efficiently loading large volumes of data into Snowflake tables using the **`COPY INTO <table>`** command.

When exporting data from Snowflake, adhering to established best practices can optimize performance, ensure data integrity, and enhance overall efficiency. The following considerations are crucial:

- **File format selection**: Carefully evaluate the intended use of the exported data to determine the most appropriate file format (CSV, Parquet, JSON) considering downstream analysis tools and compatibility requirements.
- **Storage destination**: Decide appropriate Snowflake's internal storage (stages) or external cloud storage solutions (Amazon S3, Azure Blob Storage, and so on) to unload data factoring cost, accessibility, and integration requirements.
- **Performance optimization**: For large datasets, leverage a larger virtual warehouse to expedite data unloading. Utilize the option to partitioning or splitting data into multiple files to parallelize operations.
- **Data organization and naming conventions**: Establish clear naming conventions for exported files, incorporating relevant information such as date, content description, or source table. Maintain a well-structured directory hierarchy within storage locations.

- **Compression**: Implement compression algorithms (GZIP, and so on.) to reduce storage footprint and optimize data transfer speeds. Ensure compatibility with downstream systems.
- **Security and access controls**: Implement robust access controls to safeguard exported data. Utilize encryption and role-based access to restrict unauthorized access.

By adhering to these best practices, organizations can streamline the data unloading process from Snowflake, ensuring the data's availability, integrity, and security for subsequent analysis and utilization.

Conclusion

In this chapter, you learned how the data gets loaded and unloaded to and from Snowflake. We also learned the additional technical details around the Snowflake data load/unload process, such as Stages, file formats, file size, various data loading and unloading methods, and commands. Hope the chapter summary helps the readers to summarize overall learning and keynotes to remember for the exam.

In the next chapter, we will learn about various types of data that we can store and manage in Snowflake. In addition, we will learn the technical details about how to work with the structured, semi-structured and unstructured data in Snowflake.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **What is Snowflake stage?**

  A Snowflake stage serves as a temporary cloud storage location for data files. This can be used for data loading or unloading process in Snowflake.

- **List the types of Snowflake stages?**

  Snowflake stages are categorized into the following two types.

  - **External stages**: External stages are used to store data in external cloud storage buckets. Snowflake external stages can be configured to following cloud storage providers:
    - Amazon S3
    - Microsoft Azure Blob Storage
    - Google Cloud.
  - **Internal stages**: Internal stages are used to store data files in Snowflake's own storage layer. Internal stages can be used to store data temporarily

before it is loaded into a table or to store data that is being unloaded from a table. Internal stages are further divided into three types:

- Named stages
- User stages
- Table stages

- **What is the cost model for Snowflake data transfer cost?**

  Following are the high-level data transfer cost model for Snowflake:

  - **Data Ingress**: Snowflake does not charge for data imported into your account. However, your cloud storage provider (for example, AWS, Google Cloud, or Azure) may impose egress fees when transferring data from their platform to Snowflake.
  - **Data Egress**: Snowflake applies a per-byte charge for data transferred outside of your account's region within the same cloud provider, or to a different cloud provider entirely. Data transfers within the same region are free of charge. The per-byte rate is region-dependent.
  - Scenarios Incurring Data Egress costs
    - **Data unloading**: Transferring data from Snowflake to external cloud storage (for example, S3, Google Cloud Storage, Azure Blob Storage).
    - **Data replication**: Creating a secondary database copy in a different region or cloud provider.
    - **External network access**: Connecting to external resources from within Snowflake stored procedures or user-defined functions (UDFs).
    - **External function usage**: Utilizing functions designed to facilitate data transfer to major cloud providers.
    - **Cross-cloud auto-fulfillment**: Enabling automated listing provisioning to consumers across different cloud regions.

Important note: Snowflake does not charge for data egress when using a Snowflake client or driver to retrieve query results, regardless of the regions or cloud platforms involved.

- **What is file format and list the available types of file format in Snowflake?**

  In Snowflake, a file format is a reusable object that encapsulates metadata defining the structure and layout of data files. This metadata includes attributes such as file type, compression method, field and record delimiters, row separator, and other format-specific options.

Snowflake supports a wide range of file formats for both structured and semi-structured data, including:

- **Delimited files (for example, CSV, TSV)**: Simple text files where values are separated by commas or tabs, ideal for basic tabular data.
- **JSON**: A flexible format for semi-structured data.
- **Avro, Parquet, ORC**: Columnar storage formats designed for efficient processing of large datasets.
- **XML**: A markup language used to represent hierarchical data structures.

- **Which file formats support only the data loading process?**
  - Avro, ORC, and XML file formats support just the data loading process
- **Which file formats support both the Data loading and unloading process?**
  - CSV, Parquet, and JSON file formats support both data loading and unloading process
- **What is the recommended file size for the data loading process?**

  Snowflake supports files of any size, but it is recommended to keep file sizes between 100 MB and 250 MB for optimal performance. If you need to load a very large file, consider splitting it into smaller files before loading.
- **List the available type of data loading process supported by Snowflake?**

  Snowflake offers comprehensive solutions for data loading, catering to diverse needs and data volumes. The optimal approach depends on specific requirements, such as data size and frequency of updates. Following are the two types of data load methods:
  - **Bulk loading:** This method leverages the COPY command to efficiently load large batches of data from files residing in cloud storage or local machine. It supports simple transformations like column reordering, omission, casting, and truncating text string that surpass the target table column etc. during the loading process, streamlining data preparation.
  - **Continuous loading:** This process utilizes the continuous load using Snowpipe. Designed for real-time data ingestion, this approach seamlessly integrates with various data sources, ensuring Snowflake tables are consistently up-to-date. It is particularly well-suited for scenarios where data is generated continuously.

- **What is Snowpipe and functionality of Snowpipe?**

  Snowpipe is a feature within Snowflake designed for continuous, near-real-time data ingestion. It efficiently loads micro-batches of data, making them available for analysis within minutes of arrival.

- **What is cost model for Snowpipe?**

  Snowflake's Snowpipe is a serverless data loading service that automatically scales compute resources based on your needs. This means you only pay for what you actually use, with no need to provision or manage virtual warehouses.

  Compute Resources: Snowflake charges for the compute resources consumed during data loading. This is calculated based on a per-second, per-core basis, and the usage is translated into Snowflake credits.

Chapter 11
Snowflake Data Transformations

Introduction

In this chapter, we will learn about the diverse data landscape within Snowflake, equipping you with the knowledge and skills to effectively store and manage various data types. We will explore the particulars of working especially with semi-structured, and unstructured data, empowering users to harness the full potential of Snowflake's data storage capabilities. Additionally, we will shed light on the significance of **User-Defined Functions** (**UDFs**) and Stored Procedures, unveiling their role in streamlining data manipulation and automation within the platform.

Structure

In this chapter, we will cover the following topics:

- Snowflake: Various types of data
- Supported file formats, data types
- Working with semi-structured data
- Working with unstructured data
- User-defined functions
- Stored procedures

Objectives

In this chapter, the readers will gain knowledge on various available data types in Snowflake, how to work with structured, semi-structured, and unstructured data, and other technical functionalities using UDF and stored procedures. By the end of this chapter, readers will possess a comprehensive understanding of data storage and manipulation techniques within Snowflake, enabling users to optimize their data management practices.

Snowflake: Various types of data

Snowflake offers a comprehensive suite of data types to cater to diverse data storage and manipulation needs. These include numbers, text, dates, semi-structured, and geographic data. This extensive data type support empowers Snowflake to serve as a versatile platform for storing, managing, and analyzing a wide spectrum of information, from basic records to complex scientific datasets. Refer to the following section for the various Snowflake data types to meet the industry demand.

Numeric data types

These data types stores numbers, such as integers, decimals, and floating-point numbers. Examples of numeric data types in Snowflake include:

- **`NUMBER`**: This data type stores fixed-point numbers, such as 1, 2, 3, and -100. The default precision is 38 and the default scale is 0.
- **`FLOAT`**: This data type stores floating-point numbers, such as 3.14159 and -1.234567e8. The default precision is 24.
- **`DECIMAL`**: This data type stores fixed-point numbers, such as 123.456 and -999.99. with a specified precision and scale.

String and binary data types

These data types store text and binary data. Examples of string and binary data types in Snowflake include:

- **`STRING`**: This data type stores variable-length text strings. The default length is **`VARCHAR(16,777,216)`**.
- **`BINARY`**: This data type stores variable-length binary strings, such as images and videos. The default length is **`VARBINARY(16,777,216)`**.

Logical data types

These data types store Boolean values. Examples of logical data types in Snowflake include:

- **`BOOLEAN`**: This data type stores Boolean values, which are either **`TRUE`** or **`FALSE`**.

Date and time data types

These data types store dates and times. Examples of date and time data types in Snowflake include:

- **`DATE`**: This data type stores date values in the format of YYYY-MM-DD.
- **`TIME`**: This data type stores times in the format of HH:MM:SS.
- **`TIMESTAMP`**: This data type stores timestamps in the format of YYYY-MM-DD HH:MM:SS.

Snowflake also supports a variety of other data types, such as timestamps with time zones. This makes it a popular choice for any type of business.

- **`TIMESTAMP_LTZ` (Timestamp with Local Time Zone)**: Stores time in UTC internally but adjusts to the current session's time zone for all operations. This ensures an accurate representation of the time relative to the user's location.
- **`TIMESTAMP_NTZ` (Timestamp without Time Zone)**: Stores *wall-clock* time without any time zone reference. It represents the exact time of an event, regardless of geographical location.
- **`TIMESTAMP_TZ` (Timestamp with Time Zone)**: Stores UTC time along with its associated time zone offset. This allows for precise tracking of events across different time zones

When comparing **`TIMESTAMP_TZ`** values, Snowflake uses their equivalent UTC times to ensure consistent and accurate results, irrespective of the original time zones. This approach facilitates seamless data analysis and reporting across diverse geographical locations.

Semi-structured data types

This data type stores semi-structured data, such as JSON and XML. A **`VARIANT`** can have a maximum size of up to 16 MB of uncompressed data; and can also store other data types like **`OBJECT`** and **`ARRAY`**. Examples of semi-structured data types in Snowflake include:

- **`VARIANT`**: This data type stores JSON and XML data. Example below :
  - JSON: `{ "name": "John Doe", "age": 30 }`
  - XML: `<employee><name>John Doe</name><age>30</age></employee>`

Structured data types

Snowflake supports the following structured data types for organizing complex information:

- **`ARRAY`**: A collection of **`INTEGER`** elements.

- **OBJECT**: A collection of key-value pairs (**VARCH**AR and **NUMBER**)
- **MAP**: This associates a **VARCHAR** key with a **DOUBLE** values

A structured type of column supports up to a maximum of 1000 sub-columns. Casting between semi-structured types (**ARRAY**, **OBJECT**, **VARIANT**) and their corresponding structured types is supported. For instance, an **ARRAY** can be cast to a structured type representing an **ARRAY** of **INTEGER** elements. The reverse operation, casting from a structured type to a semi-structured type, is also possible.

Geospatial data types

These data types store geospatial data, such as points, lines, and polygons on the earth's surface. Examples of geospatial data types in Snowflake include:

- **GEOGRAPHY**: This data type stores geospatial data in the WGS-84 coordinate system.
- **GEOMETRY**: This data type stores geospatial data in the EPSG:4326 coordinate system.

  This includes latitude and longitude coordinates. For example:

  - **POINT**: A point on a map, such as (37.7833, -122.4167).
  - **LINESTRING**: A line on a map, such as ((37.7833, -122.4167), (37.7933, -122.4267)).
  - **POLYGON**: A polygon on a map, such as ((37.7833, -122.4167), (37.7933, -122.4267), (37.7833, -122.4367), (37.7733, -122.4267), (37.7833, -122.4167)).

Vector data types

Snowflake has a special data type called **VECTOR**. This **VECTOR** data type allows it to handle complex numerical data (vectors) in a fast and efficient way. This is particularly useful for applications that rely on understanding the meaning behind the text, like **RAG** (**Retrieval Augmented Generation**), or common procedures on vectors in vector-based applications.

Unsupported data types

Snowflake does not currently offer direct support for the following data types:

- Large Objects (LOBs):
  - **BLOB (Binary Large Object)**: As an alternative, consider using the **BINARY** data type, which supports a maximum size of 8MB.
  - **CLOB (Character Large Object)**: The **VARCHAR** data type can be used as a substitute, accommodating up to 16MB for single-byte characters.
- Others: ENUM (Enumerated type)
- User-defined data types

Supported file formats, and file sizes

Snowflake's support for a wide variety of file formats makes it easy to load and unload data from a variety of sources and destinations. The Snowflake cloud data warehouse supports the following file formats for loading and unloading data:

- **Delimited file formats**: In Snowflake, delimited file formats refer to any file format where characters separate individual fields (columns) within each record (row). These characters, called delimiters, for example:
    - Comma-separated values (CSV) - commas ("**,**")
    - Tab-separated values (TSV) - tabs ("**\t**")
    - Pipe-separated values (PSV) - pipes ("**|**")
    - Other special characters
- **Semi-structured file formats**: Snowflake natively supports semi-structured data, which includes the following:
    - **JSON**: Any plain text file consisting of one or more JSON documents (objects, arrays, and so on).
    - **XML**: Any plain text files conforming to the XML standard.
- **Open File formats**: Snowflake natively supports the following open file formats.
    - **Avro**: A binary file format that is column-oriented and compressed.
    - **ORC**: A binary file format that is column-oriented and compressed.
    - **Parquet**: A binary file format that is column-oriented and compressed.

Modern enterprises collect vast amounts of semi-structured data from diverse sources, including weblogs, social media feeds, and sensor networks. This data holds valuable insights, yet traditional data management solutions often struggle to effectively process and analyze it alongside structured data. Snowflake offers a compelling solution, enabling seamless integration and analysis of both structured and semi-structured data within a unified cloud-data platform.

Working with semi-structured data

Semi-structured data is a form of data that does not adhere to the rigid tabular format of relational databases. However, it contains tags or other markers that distinguish individual elements and create a hierarchical structure. Two key characteristics differentiate semi-structured data from structured data:

- **Schema flexibility**: Unlike structured data, which requires a predefined schema, semi-structured data does not require a fixed schema and can evolve organically, accommodating new attributes as needed.

- **Hierarchical structure**: While structured data represents information in a flat table, semi-structured data can encompass nested hierarchies of information, enabling the representation of complex relationships.

Furthermore, entities within the same class in semi-structured data may possess varying attributes despite their grouping, and the order of these attributes is not significant.

Loading semi-structured data into Snowflake

Snowflake offers comprehensive support for semi-structured data, facilitating seamless ingestion and management of diverse formats such as JSON, Avro, ORC, Parquet, and XML. Leveraging specialized data types, Snowflake optimizes storage efficiency, enabling users to store semi-structured data in single or multiple columns based on data characteristics and individual requirements.

The loading process for semi-structured data aligns with that of structured data while providing enhanced flexibility in schema definition. Key-value pairs can be ingested into **`OBJECT`** columns, arrays into **`ARRAY`** columns, and hierarchical data can be strategically distributed across multiple columns or consolidated within a single **`VARIANT`** column.

Snowflake supports loading semi-structured data in several ways. The most common way is to use the **`COPY INTO`** statement. To load semi-structured data using **`COPY INTO`**, you will need to create a table with a column of type **`VARIANT`**. The **`VARIANT`** data type can store any type of data, including semi-structured data.

Data size limitations

The **`VARIANT`** data type in Snowflake can accommodate up to 16 MB of uncompressed data. However, due to internal overhead and the nature of the stored object, the practical limit is often lower.

In general, JSON datasets consist of simple concatenation of multiple documents. Some software may output JSON as a single large array containing multiple records. While line breaks or commas are optional for separating documents, both formats are supported.

If the data size exceeds 16 MB, consider utilizing the **`STRIP_OUTER_ARRAY`** file format option with the **`COPY INTO <table>`** command. This removes the outer array structure, enabling each record to be loaded into a separate table row.

Querying semi-structured data in Snowflake

Snowflake supports the **`FLATTEN`** table function, which can be used to convert the semi-structured data to a relational format. **`FLATTEN`** is a table function designed to deconstruct complex data structures within **`VARIANT`**, **`OBJECT`**, or **`ARRAY`** columns. It generates a lateral view, essentially a virtual table, that can be seamlessly integrated with existing tables in the

query's **FROM** clause. This functionality facilitates the transformation of semi-structured data into a relational format, enabling efficient analysis and manipulation of complex datasets.

Snowflake's **LATERAL FLATTEN** function is another table function for working with complex, nested data like JSON files or arrays. **LATERAL FLATTEN** operates by recursively traversing nested data hierarchies, extracting values at each level, and generating a new row for each extracted element. The inclusion of the **LATERAL** keyword signifies that the function produces a lateral view, facilitating seamless joins with other tables or views within the same query.

While both **LATERAL FLATTEN** and **FLATTEN** functions can unpack nested data, the key distinction lies in the former's ability to perform lateral joins. This means that the resulting rows from **LATERAL FLATTEN** can be seamlessly joined with other tables in the database, significantly expanding the scope of analysis.

For instance, consider a table containing book information with an array column storing multiple authors per book. **LATERAL FLATTEN** can deconstruct this array, generating a row for each author while preserving the association with their respective book. This transformation facilitates querying tasks such as identifying all authors in the database or retrieving the complete works of a specific author.

In summary, **FLATTEN** and **LATERAL FLATTEN** enhances Snowflake's capability to manipulate complex data structures by providing a streamlined mechanism for data extraction and integration within a relational framework, thereby empowering analysts to derive valuable insights efficiently.

Working with unstructured data

Unstructured data, information that does not conform to a predefined data model, encompasses diverse content types including text-heavy documents (for example, form responses, social media conversations), multimedia (images, video, audio), and industry-specific file formats (for example, VCF, KDF, HDF5).

Snowflake offers robust capabilities for managing unstructured data:

- **Secure data access**: Provides secure access to data files residing in cloud storage, ensuring data confidentiality and integrity.
- **Enhanced collaboration**: Facilitates seamless sharing of file access URLs with internal teams and external partners, promoting efficient collaboration.
- **Metadata management**: Enables loading of file access URLs and associated metadata into Snowflake tables, facilitating efficient organization and tracking of unstructured data assets.
- **Data processing**: Supports comprehensive processing and analysis of unstructured data within the Snowflake environment, unlocking valuable insights and enabling data-driven decision-making.

Snowflake offers a robust and versatile solution for managing unstructured data. Leveraging both external and internal stages, Snowflake seamlessly integrates with major cloud storage providers like Amazon S3, Google Cloud Storage, and Microsoft Azure (Blob storage, Data Lake Storage Gen2, General-purpose v1/v2) for scalable, flexible external storage. Alternatively, organizations can opt for Snowflake's secure, managed internal storage for streamlined data access within the platform.

Cloud storage facilitates file access through the following URL types:

- **Scoped URLs**: These encoded URLs provide temporary, time-limited access to staged files without granting additional permissions to the stage itself. The scope of the URL typically expires within 24 hours when the persisted query result or cache expires in Snowflake.
- **File URLs**: These URLs directly reference the file's location within the cloud storage structure (database, schema, stage, and file path). Access is contingent upon the requesting user's role possessing sufficient privileges on the specified stage.
- **Pre-signed URLs**: Utilizing pre-signed access tokens, these standard HTTPS URLs enable temporary file access through web browsers. The token's expiration time is configurable, affording administrators flexibility in managing access durations.

Processing unstructured data

Snowflake offers a robust suite of capabilities to facilitate the processing of unstructured data:

- **External functions**: Leverage user-defined functions executed outside Snowflake to integrate with external libraries like Amazon Textract or Google Document AI, enhancing your ability to extract insights from diverse unstructured data sources.
- **Customizable functions and procedures**: Extend Snowflake's SQL capabilities or develop bespoke applications using the Snowpark API, enabling the creation of tailored functions and procedures in Java or Python to process unstructured data or incorporate proprietary machine learning models.
- **Snowpark API**: Build scalable and efficient data pipelines and applications that seamlessly interface with Snowflake's data processing engine, empowering you to manage and analyze unstructured data in a flexible and integrated manner.

These comprehensive provisions enable organizations to effectively harness the value of unstructured data within the Snowflake Data Cloud, driving data-driven insights and informed decision-making at scale.

Here are some best practices for working with unstructured data in Snowflake:

- **Leverage external stages**: Employ external stages as repositories for unstructured data. This approach preserves the native format of the data, eliminating the need for conversion and simplifying data ingestion.

- **Utilize UDFs (User-Defined Functions)**: Develop custom UDFs to perform targeted operations on unstructured data. This includes tasks such as extracting relevant text, identifying entities, and categorizing data elements, all within the Snowflake environment.
- **Integrate with external tools**: For complex analytical tasks, seamlessly integrate Snowflake with specialized external tools. This facilitates advanced operations like training and deploying machine learning models to uncover patterns and insights within unstructured data.
- **implement data tagging**: Apply metadata tags to unstructured data to enhance organization and searchability. Tagging streamlines the process of locating specific data and provides a clearer understanding of the data landscape.

Snowflake's inherent features significantly streamline the management of unstructured data. By implementing these best practices, organizations can effectively store, process, and analyze unstructured data, there by extracting valuable insights and driving informed decision-making:

- Use external stages to store unstructured data. This will allow you to store unstructured data in its native format and avoid the need to convert it to a structured format.
- Use UDFs to process and analyze unstructured data. UDFs can be used to perform a wide variety of tasks on unstructured data, such as extracting text, identifying entities, and classifying data.
- Use external tools to process and analyze unstructured data. External tools can be used to perform more complex tasks on unstructured data, such as training and deploying machine learning models.
- Use data tagging to organize and manage your unstructured data. Data tagging allows you to add metadata to your unstructured data, which can make it easier to find and analyze.

Snowflake's inherent features significantly streamline the management of unstructured data. By implementing these best practices, organizations can unlock the full potential of their unstructured data assets, driving data-driven strategies and achieving business objectives.

User-defined functions

A **user-defined function** (**UDF**) is a custom function created to extend the capabilities of SQL queries. Like built-in SQL functions, UDFs allow developers to incorporate logic not natively available or optimized within standard SQL.

In Snowflake, UDFs play a crucial role in enabling operations not supported by Snowflake's pre-built functions. This flexibility empowers developers to tailor the system to their

specific requirements. The UDFs can be called from an SQL repeatedly from multiple places in a code and can be used to extend the functionality of Snowflake or to encapsulate complex logic that can be reused throughout your data pipeline.

Snowflake supports the creating the UDFs in one of the following programming languages:

- SQL
- JavaScript
- Java
- Python
- Scala

The creation of a UDF involves writing its core logic, referred to as the handler, written in one of the compatible programming languages. To create a UDF, you use the **CREATE FUNCTION** statement. The **CREATE FUNCTION** statement specifies the name of the UDF, the language in which it is written, and the arguments that it takes.

Once you have created a UDF, you can call it from SQL via a standard **SELECT** statement just like any other function. Refer to the following section for the sample code for creating UDF's using SQL and Python languages. The users can choose the appropriate language based on their organization or users preference:

```
SQL:
--Example 1:
-- Sample Syntax/Code to create SQL UDF for converting to Upper case:
CREATE OR REPLACE FUNCTION convert_uppercase (input VARCHAR)
RETURNS VARCHAR
AS
$$
  SELECT UPPER(input)
$$
;
-- Call the convert_uppercase UDF using the SELECT statement
SELECT convert_uppercase('david');
-- Output:
This would return the value as DAVID.
--Example 2:
-- Create a UDF that takes two integer values and returns the sum values
CREATE OR REPLACE FUNCTION add_two_numbers (a number, b number)
  RETURNS number
  COMMENT='Add two numbers'
  AS 'a + b';
```

```
-- Call the add_two_numbers UDF using the SELECT statement
SELECT add_two_numbers(5, 6);
-- Output:
This would return the value as 12.
--Example 3:
-- Create a UDF using Python
CREATE OR REPLACE FUNCTION add_one(i int)
RETURNS INT
LANGUAGE PYTHON
RUNTIME_VERSION = '3.8'
HANDLER = 'addone_py'
as
$$
def addone_py(i):
  return i+1
$$;
-- Call the UDF using the SELECT statement
SELECT add_one(5);
-- Output:
This would return the value 6.
```

UDFs are a powerful feature for extending the functionality of Snowflake and encapsulating complex logic. By following the best practices outlined in this section can help organizations leverage Snowflake UDFs to extend the data warehouse capabilities and maximize the value of the data assets.

The following are the best practices for working with UDFs in Snowflake:

- **Strategic language selection**: Choose the appropriate programming language (JavaScript, Java, Python, or SQL) based on your team's skill set and the specific requirements of the UDF. Evaluate each language's strengths and weaknesses concerning performance, maintainability, and ease of integration.
- **Modular design**: Decompose complex UDFs into smaller, reusable modules to enhance code readability, testability, and maintainability. This approach promotes collaboration, simplifies debugging, and reduces the risk of errors.
- **Robust security measures**: Adhere to Snowflake's security best practices to protect sensitive data.
- **Performance optimization**: Design UDFs with performance in mind. Utilize efficient algorithms, minimize memory consumption, and leverage Snowflake's parallel processing capabilities where applicable. Monitor UDF execution times and optimize as needed.

- **Thorough testing**: Develop a comprehensive testing strategy to validate UDF functionality and performance across a wide range of input values and scenarios. Include edge cases, error conditions, etc. to ensure robustness.
- **Comprehensive documentation**: Document UDFs thoroughly, including purpose, input parameters, return values, error handling mechanisms, and usage examples. This documentation facilitates understanding, collaboration, and future maintenance.

Following are a few important considerations for using the UDFs in Snowflake:

- **Performance**: SQL UDFs are generally the fastest for simple tasks, while Java and Scala can offer better performance for complex computations. JavaScript and Python can be suitable for mid-level complexity.
- **Ease of development**: SQL UDFs are the easiest to write for simple tasks. JavaScript is also relatively easy to pick up. Java and Scala require more specialized knowledge. Python (with Snowpark) is a good option for data scientists familiar with Python.
- **Sharing**: SQL and JavaScript UDFs are easily shareable across Snowflake accounts, while Java, Scala, and Python UDFs are not.
- **Snowpark**: A newer framework within Snowflake that allows you to use Java, Scala, or Python to interact with Snowflake data using familiar DataFrame concepts. Offers more flexibility and integration with the broader data science ecosystem.

Stored procedure

A Snowflake stored procedure is a named set of SQL statements that can be executed as a single unit. Stored procedures can be used to encapsulate complex logic, to automate tasks, improve performance, and simplify application development.

To create stored procedures, use the **`CREATE PROCEDURE`** statement. The **`CREATE PROCEDURE`** statement specifies the procedure's name, parameters, and handler code.

Snowflake stored procedures can be written in one of the following languages:

- SQL
- Java (using Snowpark)
- JavaScript
- Python (using Snowpark)
- Scala (using Snowpark)
- Snowflake Scripting

Snowflake stored procedures offer flexibility in development by supporting multiple programming languages, including Java, JavaScript, Python, Scala, and Snowflake Scripting (SQL). Each language adheres to its own syntax and operational constraints, but all procedures are uniformly created using SQL, specifying the chosen language and the location of the handler code.

Selecting the appropriate language involves evaluating several factors. If pre-existing code aligns with a specific language, leveraging it can streamline development. Additionally, certain languages may possess unique capabilities or libraries that cater to the specific requirements of the procedure. Furthermore, the designated storage location for the handler code, whether inline or staged, can influence the language choice, as not all languages support both options.

Refer to the below section for the sample code for creating the SQL and Python Stored procedure. The users can choose the appropriate language based on their organization or users' preferences. Once you have created a stored procedure, you can use it for multiple times. Executing Stored procedure using the SQL **`CALL`** statement.

```
SQL:
-- Sample Syntax/Code to create Stored Procedure using SQL Language:
CREATE OR REPLACE PROCEDURE return_greater_number(number_1 INTEGER,
number_2 INTEGER)
RETURNS INTEGER NOT NULL
LANGUAGE SQL
AS
$$
BEGIN
  IF (number_1 > number_2) THEN
    RETURN number_1;
  ELSE
    RETURN number_2;
  END IF;
END;
$$
;
-- Sample Syntax/Code to execute/call the Stored Procedure:
CALL return_greater_number(100, 500);
-- Output:
This would return the value 500
-- Sample Syntax/Code to create Stored Procedure using Python:
create or replace procedure data_load(from_table string, to_table string,
count int)
  returns string
```

```
  language python
  runtime_version = '3.8'
  packages = ('snowflake-snowpark-python')
  handler = 'run'
as
$$
def run(session, from_table, to_table, count):
  session.table(from_table).limit(count).write.save_as_table(to_table)
  return "SUCCESS"
$$;
-- Sample Syntax/Code to execute/call the Stored Procedure:
CALL data_load('customer_s', 'customer_t', 5);
```

In conclusion, Snowflake's multi-language support for stored procedures empowers developers to select the most suitable language based on their existing codebase, the specific functionalities required, available libraries, and the desired storage location for the handler code. This adaptability contributes to the efficient and effective development of stored procedures within the Snowflake environment.

There are several benefits to using stored procedures in Snowflake, including:

- **Improved performance**: Stored procedures can improve the performance of your code by compiling the SQL statements in the procedure once and then executing them multiple times.
- **Reduced code duplication**: Stored procedures can help to reduce code duplication by encapsulating common logic in a single procedure. This can make the code easier to maintain and reusable.
- **Improved modularity**: Stored procedures can help to improve the modularity of your code by breaking it down into smaller, more manageable units.

Here are some best practices for working with stored procedures in Snowflake:

Snowflake stored procedures offer a powerful mechanism for encapsulating and automating complex data processing logic within the Snowflake data cloud. To leverage their full potential and ensure optimal performance, consider the following best practices:

- **Modular development**: Improving the modularity of the complex stored procedure breaking it down into smaller, more manageable, and reusable code. This approach enhances code maintainability, readability, and testability, streamlining future modifications.
- **Robust error handling**: Incorporate comprehensive error handling mechanisms using Snowflake's **`TRY...CATCH`** blocks. This ensures graceful handling of unexpected scenarios, preserves data integrity, and enables proactive troubleshooting.

- **Transactional integrity**: Utilize transactions to guarantee atomicity (all-or-nothing execution) for sequences of SQL statements within stored procedures. This prevents partial updates and maintains data consistency in case of errors.
- **Parameterization**: Parameterize your stored procedures to promote reusability and adaptability to varying input data. Avoid hardcoding values whenever possible to maintain flexibility.
- **Stringent security**: Adhere to the principle of least privilege when granting permissions for stored procedure execution. Explore *owner's rights* stored procedures for sensitive operations and implement Snowflake's security best practices for **user-defined functions** (**UDFs**).
- **Comprehensive logging**: Maintain detailed logs of procedure execution, capturing relevant information such as procedure names, input parameters, execution durations, and error messages. This facilitates debugging and auditing processes.
- **Version Control Integration**: Utilize version control systems (for example, Git) to track changes to your stored procedure codebase. This enables historical review, facilitates rollbacks when necessary, and provides a comprehensive audit trail.
- **Rigorous testing**: Develop a thorough testing strategy encompassing unit tests, integration tests, and mock external dependencies. This ensures the accuracy, robustness, and reliability of your stored procedure logic.
- **Standardized naming**: Adopt a consistent naming convention for your stored procedures to improve code organization and maintainability.
- **Performance optimization**: Utilize Snowflake's **`RESULT_SCAN`** function to analyze the query plan of your stored procedures, identifying potential performance bottlenecks and opportunities for optimization.

By adhering to these best practices and remaining mindful of the inherent limitations, you can harness the full power of Snowflake stored procedures to streamline data operations, automate complex processes, and drive efficiency within your data ecosystem.

Conclusion

In this chapter, we learned about various types of data that we can store and manage in Snowflake. In addition, we also learned the technical details about how to work with the structured, semi-structured, and unstructured data in Snowflake and the importance of user-defined functions and stored procedures in Snowflake. Hope the chapter summary helps the readers to summarize overall learning for the exam.

In the next chapter, we will learn about how to keep all the information safely secured in Snowflake using various available data protection features in Snowflake such as time travel, fail-safe, data encryption, cloning, replications, and so on.

Points to remember

Refer to the following summary for a quick reference to what we learned so far in this chapter:

- **List the various supported and non-supported data types in Snowflake:**

  Snowflake supports a wide range of data types to handle various structured, semi-structured, and geography data types, the following are the supported data types by Snowflake:

  - Numeric:
    - **`NUMBER`** (with precision and scale, for example, **`NUMBER(38,2)`**)
    - **`DECIMAL`**, **`NUMERIC`** (synonyms for **`NUMBER`**)
    - **`INT`**, **`INTEGER`**, **`BIGINT`**, **`SMALLINT`**, **`TINYINT`**, **`BYTEINT (integers)`**
    - **`FLOAT`** (floating-point numbers)
  - Character/string:
    - **`VARCHAR`**, **`STRING`**, **`TEXT`** (variable-length strings)
    - **`CHAR`**, **`CHARACTER`** (fixed-length strings)
    - **`BINARY`** (binary data)
  - Date and time:
    - **`DATE`**
    - **`TIME`**
    - **`TIMESTAMP, TIMESTAMP_NTZ (without time zone)`**
    - **`TIMESTAMP_TZ, TIMESTAMP_LTZ (with time zone)`**
  - Boolean: BOOLEAN
  - Semi-structured:
    - **`VARIANT`** (can store various data types, including objects and arrays)
    - **`OBJECT`** (key-value pairs)
    - **`ARRAY`**
  - Geographic:
    - **`GEOGRAPHY`**
    - **`GEOMETRY`**
  - Other: UUID (Universally Unique Identifier)

The following are the unsupported data types by Snowflake:

- Large Objects (LOBs): BLOB, CLOB (use **`BINARY`** and **`VARCHAR`** instead)
- **`ENUM`** (enumerated types)
- User-Defined Data Types

- **What is the File format and list the support file format by Snowflake?**

  In Snowflake, a file format is a reusable object that encapsulates the metadata defining the structure and layout of data files. This metadata includes attributes such as file type, compression method, field and record delimiters, row separator, and other format-specific options

  Snowflake supports a variety of file formats for both structured and semi-structured data, including:

  - **Delimited files (for example, CSV, TSV)**: Simple text files where values are separated by commas or tabs. Great for basic, tabular data.
  - **JSON**: A flexible format for semi-structured data that can be easily read by humans.
  - **Avro, Parquet, and ORC**: Columnar storage formats designed for efficient processing of large datasets.
  - **XML**: A markup language used to represent hierarchical data structures.

- **What is the Stored procedure in Snowflake?**

  A stored procedure in Snowflake is a reusable block of code that can be called within SQL queries. It's designed to perform database operations by executing multiple SQL statements together, often with some additional procedural logic for easier management and execution.

- **List the supported language and key benefits of using the Stord procedure in Snowflake.**

  Snowflake empowers developers with a versatile selection of programming languages for crafting stored procedures:

  - **SQL**: Primarily used for straightforward data manipulation and basic logic execution.
  - **JavaScript**: Ideal for intricate control flow, complex calculations, and seamless integration with external APIs.
  - **Java (Snowpark)**: Leverages existing Java libraries and expertise for advanced data processing and manipulation.
  - **Python (Snowpark)**: Renowned for its readability, vast data science libraries, and streamlined scripting capabilities.
  - **Scala (Snowpark)**: Combines functional and object-oriented programming paradigms for sophisticated data transformations and analysis.

  Key benefits of using stored procedures in Snowflake:

  - **Modularity and reusability**: Procedures can be called multiple times, promoting code organization and reducing redundancy.

- **Improved performance**: Procedures can reduce network traffic by minimizing the number of round trips to the database server.
- **Centralized business logic**: Procedures provide a central location for common operations, making code maintenance easier.
- **Enhanced security**: Procedures can be executed with the privileges of their owner, not the caller, providing better access control.
- **Dynamic SQL**: Procedures can generate and execute SQL statements on the fly, adding flexibility.
- **Simplified client applications**: Client applications can call procedures instead of complex SQL, making them simpler and easier to maintain.

- **What is the User Defined Function(UDF) in Snowflake and supported languages?**

  In Snowflake, a UDF is a custom function you create to perform specific tasks. UDFs help to extend Snowflake's capabilities beyond its built-in functions, making your data processing and analysis more efficient.

  Snowflake supports a variety of programming languages for UDF development, including:

  - SQL
  - Java
  - JavaScript
  - Python
  - Scala

- **List the few key benefits of using the UDF in Snowflake.**

  Key benefits of leveraging UDFs in Snowflake include:

  - **Streamlined code maintenance**: Centralized function definitions enhance code organization and simplify updates.
  - **Improved code structure**: Modularizing complex logic into UDFs fosters cleaner, more maintainable SQL code.
  - **Performance optimization**: Utilize language-specific strengths, such as Java's speed, for demanding computations.
  - **Expanded functionality**: Tailor Snowflake to unique business requirements by implementing custom logic.
  - **Increased productivity**: Accelerate development cycles by reusing pre-built functions.

Chapter 12
Snowflake Data Protection

Introduction

In this chapter, we will discuss how Snowflake keeps your data safe and secure. We will explore features like time travel and fail-safe that let you recover accidentally deleted or modified data. We will also learn about encryption that secures your information, making it unreadable to unauthorized users. Additionally, we will cover cloning and replication options for creating secure copies of your data.

Structure

In this chapter, we will cover the following topics:

- Data protection overview
- Snowflake Time Travel
- Snowflake Fail-safe
- Data encryption
- Cloning
- Replication

Objectives

This chapter aims to provide comprehensive information about various Data Protection features available in Snowflake such as Time Travel, Fail-safe, data encryption, cloning, replications, and so on. By the end of this chapter, readers will gain knowledge of these data protection techniques and also learn about how to keep the information secured and safe within Snowflake using these features.

Data protection overview

Snowflake Data Protection is a robust suite of features designed to safeguard organizational data stored within Snowflake's cloud-based environment. It provides a comprehensive approach to data protection by addressing a wide spectrum of threats, including human error, malicious acts, and software failures.

Key features of Snowflake data protection

Following are some key features of Snowflake data protection:

- **Encryption**: Snowflake utilizes industry-standard encryption algorithms to encrypt all data at rest and in transit, ensuring Data Protection even in the event of underlying infrastructure compromise.
- **Access control**: Snowflake's granular **role-based access control** (**RBAC**) mechanism allows organizations to precisely control user access to specific data objects based on their roles and permissions, safeguarding sensitive data from unauthorized access.
- **Auditing**: Snowflake's comprehensive auditing capabilities enable organizations to track data access activities, including who accessed the data and when. This valuable information facilitates incident investigations and compliance with Data Protection regulations.
- **Time travel**: Snowflake's unique time travel feature empowers organizations to restore data to a previous point in time, even if it has been deleted or modified, providing a safety net against accidental data loss or corruption.
- **Fail-safe:** Snowflake fail-safe protects your data from accidental deletion or corruption for an additional seven days of data recovery by Snowflake that starts immediately after the Time Travel period ends.
- **Row-level security**: Snowflake's row-level security feature allows organizations to restrict access to specific rows within a table based on user permissions, further safeguarding sensitive data such as customer **Personally Identifiable Information** (**PII**) or financial data.
- **Replication**: Snowflake proactively replicates data across multiple availability zones within a region, ensuring data availability and accessibility even if one availability zone encounters disruptions.

- **Disaster recovery**: Snowflake offers a range of disaster recovery options, including cross-region replication and backup to object storage, ensuring data protection in the event of major disasters such as natural calamities or cyber-attacks.

Benefits of Snowflake data protection

Organizations of all sizes can reap significant benefits from implementing Snowflake Data Protection:

- **Enhanced data security**: Snowflake Data Protection's comprehensive security measures effectively safeguard data against a wide range of threats, ensuring data integrity and confidentiality.
- **Improved compliance**: It facilitates compliance with severe Data Protection regulations, such as **Health Insurance Portability and Accountability Act** (**HIPAA**), **General Data Protection Regulation** (**GDPR**), and **California Consumer Privacy Act** (**CCPA**), reducing compliance risks and legal liabilities.
- **Reduced data loss**: Snowflake data protection's robust data protection mechanisms minimize data loss incidents, ensuring data availability and business continuity.
- **Enhanced data governance**: It empowers organizations to implement effective data governance practices, promoting data integrity, accessibility, and control.
- **Reduced costs**: Snowflake data protection's proactive approach to Data Protection minimizes the costs associated with data breaches and compliance violations.

Snowflake data protection is a valuable asset for data-driven organizations. In today's data-driven world, data protection is paramount. Snowflake data protection provides organizations with comprehensive powerful features to safeguard their valuable data assets, ensuring data security, regulatory compliance, and business continuity. By implementing Snowflake Data Protection, organizations can confidently leverage their data to drive innovation and competitive advantage.

Snowflake data protection: Organizational impact

Snowflake Data Protection can help organizations of all sizes to protect their data and comply with Data Protection regulations. Here are a few specific examples:

- **Healthcare organizations**: Snowflake Data Protection can help healthcare organizations to protect PHI, which is subject to strict privacy regulations.
- **Financial services organizations**: Snowflake Data Protection can help financial services organizations to protect customer financial data, which is also subject to strict privacy regulations.
- **Government agencies:** Snowflake Data Protection can help government agencies protect sensitive data, such as citizen information and national security data.

Snowflake Time Travel

In today's data-driven world, organizations rely heavily on their data assets to drive business decisions and maintain a competitive edge. However, data is also vulnerable to a range of threats, including human error, malicious acts, and software failures. Data loss can have severe consequences, leading to financial losses, reputational damage, and regulatory penalties.

Snowflake Time Travel is a remarkable feature that empowers organizations to safeguard their valuable data by providing a robust mechanism to restore data to any point in time, even if it has been deleted or modified. This feature offers a safety net against data loss incidents and ensures business continuity.

This remarkable capability stems from Snowflake's unique architecture, which maintains immutable snapshots of data at regular intervals. These snapshots serve as anchors, allowing users to effortlessly retrace their data's evolution and restore it to a desired state ranging from one to 90 days.

Key features and benefits of Snowflake Time Travel

The implementation of Snowflake Time Travel offers a multitude of benefits, fostering a secure and resilient data environment:

- **Data integrity preservation**: Snowflake Time Travel safeguards data integrity by providing data loss prevention mechanism to undo the unwanted changes. This capability proves invaluable in rectifying human errors or mitigating the consequences of accidental activities.
- **Unparalleled data recovery**: Data loss, often a result of human error or system failures, can be devastating for organizations. Snowflake Time Travel acts as a lifeline, enabling the swift restoration of data to a specific point in time, minimizing downtime and business disruptions.
- **Regulatory compliance facilitation**: Adherence to severe data retention regulations is crucial for organizations operating in various industries. Snowflake Time Travel simplifies compliance by providing a comprehensive audit trail of data modifications, ensuring that data is maintained for the mandated periods.
- **Enhanced data governance**: Snowflake Time Travel empowers organizations to establish effective data governance practices, promoting data consistency, accessibility, and control. This ultimately leads to improved data quality and informed decision-making.

Real-time example of Snowflake Time Travel

In this section, we will go through a practical example of Snowflake Time Travel, which is restoring a deleted customer record. To illustrate the practical application of Snowflake Time Travel, consider the following scenario:

- A human or data operation process accidentally deletes a customer record from a table storing customer information. This record contains crucial information, including the customer's name, contact details, and purchase history.
- Utilizing Snowflake Time Travel, user can easily restore the deleted customer record to its state before deletion. This process involves identifying the timestamp of the snapshot that precedes the deletion and initiating the restoration process. This is the most critical feature of restoring using SQL command and users can restore on their own without snowflake support.
- Once the restoration is complete, the customer record is restored to its original state, alleviating the consequences of data loss and ensuring the company maintains accurate customer information.
- Restoring data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.
- Duplicating and backing up data from key points in the past.
- Analyzing data usage/manipulation over specified periods of time

Snowflake Time Travel emerges as an indispensable feature for Data Protection and business continuity. Its ability to revert data to any point in time empowers organizations to safeguard their valuable data assets, ensuring data integrity, regulatory compliance, and business resilience. By utilizing Snowflake Time Travel, organizations can foster a resilient data environment, ensuring data availability, compliance, and informed decision-making.

Snowflake Fail-safe

In the ever-evolving realm of data management, data integrity and accessibility remain paramount concerns. Snowflake Failsafe (by Snowflake) emerges as a comprehensive data recovery solution, empowering organizations to seamlessly restore their data beyond the Time Travel feature, shielding against accidental deletions, data corruption, or malicious acts. The Fail-safe feature is non configurable and allows additional seven days of data recovery by Snowflake that starts immediately after the Time Travel period ends.

Refer to the following figure for the overall data protection lifecycle in Snowflake using Time Travel and Fail-safe methods:

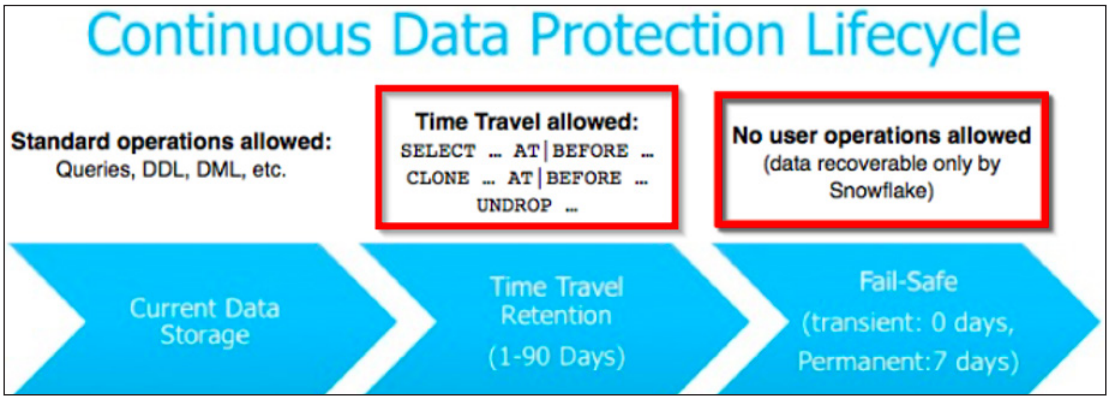

Figure 12.1: *Snowflake Time Travel and Fail-safe (Credit: Snowflake documentation)*

Real-time example of Snowflake Fail-safe

In this section, we will go through a practical example of Snowflake Fail-safe. Let's consider a scenario where an enterprise utilizes Snowflake as its primary data warehouse for storing critical business intelligence data, including financial records and customer analytics. The organization has configured Snowflake's Time Travel feature to retain historical data for a standard 30-day period, allowing for point-in-time recovery within this timeframe.

However, an unforeseen system failure results in the corruption and loss of essential data that extends beyond the 30-day retention window. In this critical situation, Snowflake's Fail-Safe mechanism becomes instrumental.

Fail-Safe, an internal data protection layer, automatically retains historical data for an additional seven days beyond the Time Travel period. Upon engaging Snowflake support, the organization can request the restoration of the lost data from the Fail-Safe repository. This enables the complete recovery of the corrupted data, ensuring business continuity and mitigating the impact of the system failure.

Key takeaways using Snowflake Fail-safe:

- **Fail-safe as a last resort**: Fail-Safe is a data recovery service of last resort, managed by Snowflake. This feature is not intended for routine data recovery. It is activated only in extreme scenarios where all other recovery options have been exhausted.
- **Non-configurable**: Fail-Safe provides a fixed seven-day retention period beyond Time Travel, and this duration cannot be altered.
- **Non-user accessible**: Fail-Safe is exclusively managed by Snowflake. Users cannot directly access or manipulate this data.
- **Enhanced data protection**: Fail-Safe serves as an additional layer of data protection, supplementing Time Travel and offering peace of mind in the face of unforeseen events.

This example highlights Fail-Safe's crucial role in safeguarding critical data and ensuring business resilience within the Snowflake environment. By incorporating Fail-Safe into their data management strategy, organizations can mitigate the risk of catastrophic data loss and ensure the continuity of their financial operations.

Snowflake Time Travel Vs Fail-Safe

This section provides the information about various key features and comparison between the Time Travel and Fail-safe methods. Refer to the following table for high-level comparisons by various features:

Feature	Time Travel	Fail-safe
Purpose	Enables querying, cloning, and restoring historical data versions within a user-defined retention period.	Provides an additional layer of data protection by facilitating data recovery from severe operational failures beyond the Time Travel retention period.
Retention Period	Configurable: Up to 90 days for Enterprise Edition (and higher), 1 day for Standard Edition.	7 days (non-configurable).
Accessibility	Directly accessible by users via SQL queries and cloning commands.	Accessible only by Snowflake Support for data recovery purposes.
Data Availability	Historical data versions are readily available throughout the retention period.	Data recovery is on a best-effort basis by Snowflake Support.
Primary Use Cases	* Auditing and compliance * Recovering from accidental data changes or deletions. * Performing historical data analysis and trend reporting.	* Disaster recovery from catastrophic events. * Protection against data corruption or loss due to unforeseen circumstances. * Last resort for data recovery when Time Travel is insufficient.
Limitations	* Requires Enterprise Edition (and higher) for retention periods exceeding 1 day. * Extended retention periods increase storage costs.	* Not intended for regular access to historical data. * Data recovery process can be time consuming. No user-level access; recovery is solely handled by Snowflake Support.

Table 12.1: *Snowflake Time Travel Vs Fail-Safe Comparison by Features*

Refer to the following high-level comparisons between the Time Travel and Fail-safe methods by different types of Snowflake tables.

Table Type	Time Travel Retention Period (Days)	Fail-safe Period (Days)	Min , Max Historical Data Maintained (Days)
Permanent	- 0 or 1 For Std Edition - 0 to 90 For Enterprise Ed and higher	7 7	0, 8 0, 97
Transient	0 or 1	NO	0, 1
Temporary	0 or 1	NO	0, 1

Table 12.2: *Snowflake Time Travel Vs Fail-Safe Comparison by Table Types*

Data encryption

In today's data-driven world, organizations are entrusted with managing vast amounts of sensitive data, including customer information, financial records, and intellectual property. Protecting this data from unauthorized access and malicious attacks is paramount. Snowflake data encryption provides a robust and comprehensive approach to safeguarding your valuable information.

Encryption at rest and in transit

Snowflake utilizes industry-standard encryption algorithms to encrypt all data at rest and in transit. Snowflake takes data security very seriously and uses strong encryption methods to protect your information both at rest (when it's stored) and in transit (when it's moving). Here's a breakdown in simpler terms:

Data at rest refers to data stored within Snowflake's infrastructure, while data in transit refers to data moving between Snowflake and your applications or users.

Encryption at rest

Data at rest refers to data stored within Snowflake's infrastructure. Here's a quick summary how it’s handled in Snowflake:

- **AES-256 encryption algorithm**: Snowflake encrypts all data at rest using the AES-256 encryption algorithm, a globally recognized standard for data security, which is considered unbreakable by current technology. This algorithm ensures that even if an attacker were to gain access to Snowflake's infrastructure, they would not be able to decrypt your data.

- **Key Management Service (KMS)**: Snowflake utilizes a KMS to securely store and manage encryption keys. This service can be either Snowflake's internal KMS or an external KMS provided by a third-party vendor, such as AWS KMS or Azure Key Vault.
- **Transparent process:** You don't need to configure or manage anything; Snowflake handles the encryption and decryption automatically.

Encryption in transit

Data at transit refers to data moving between Snowflake and your applications or users. Here's a quick summary how it's handled in Snowflake:

- **TLS/SSL encryption**: Snowflake encrypts all data in transit using TLS/SSL (Transport Layer Security/Secure Sockets Layer) protocols. These protocols ensure that data exchanged between Snowflake and your applications or users is protected from eavesdropping or interception.
- **Encrypted communication between Snowflake and clients**: Snowflake ensures that all communication between its servers and your clients, including web browsers, desktop applications, and mobile apps, is encrypted using TLS/SSL protocols. This safeguards data from unauthorized access even when accessing Snowflake from public networks.
- **HTTPS protection:** Communication between you and Snowflake always uses HTTPS, which encrypts data in transit, making it unreadable by anyone trying to intercept it

Understanding encryption key management

Snowflakes utilize a robust encryption key management strategy to safeguard customer data. By default, Snowflake manages these keys, ensuring automatic encryption and regular key rotation for enhanced security. This process is seamless and requires no customer intervention.

Snowflake automated key management, rotating all encryption keys every 30 days. Retired keys are designated for decryption purposes only, ensuring data accessibility while maintaining security. Active keys are used for encryption and remain accessible to customers throughout their lifecycle. Once retired keys are no longer required for decryption, they are securely destroyed.

Snowflake's hierarchical key model further strengthens data protection. It comprises various key layers, where each layer encrypts the layer below, isolating customer accounts and minimizing the scope of each key.

For an additional layer of security, customers can utilize the key management service on their cloud platform to manage their own encryption key. This key, combined with Snowflake's, forms a composite master key, providing even stronger protection.

Moreover, Snowflake offers a **periodic rekeying feature for Enterprise Edition (or higher)** customers. This feature further enhances security by re-encrypting data more frequently.

Snowflake's commitment to data security is evident in its multi-layered approach to encryption key management. By combining its own robust measures with the option for customer-managed keys, Snowflake empowers customers to maintain control over their data security while benefiting from the platforms advanced security features.

Tri-Secret Secure in Snowflake

Tri-Secret Secure is an advanced security feature designed for Snowflake customers utilizing **Business Critical Edition (or higher)**. By combining a Snowflake-managed key with a customer-managed key from your cloud provider, Tri-Secret Secure establishes a composite master key that significantly enhances the protection of your sensitive data within the Snowflake environment.

Important Considerations: Prior to enabling Tri-Secret Secure, thoroughly assess the responsibilities associated with managing your customer-managed key. Snowflake maintains its own keys with the utmost diligence, adhering to stringent policies that have garnered top-tier security certifications, including SOC 2 Type II, PCI-DSS, HIPAA, and HITRUST CSF.

Overall, Snowflake Data Encryption provides a comprehensive and robust approach to safeguarding your valuable information. By utilizing industry-standard encryption algorithms, granular access control mechanisms, and data masking techniques, Snowflake ensures that your data remains protected from unauthorized access, both at rest and in transit. By implementing Snowflake Data Encryption, organizations can confidently manage their sensitive data while maintaining compliance with data privacy regulations.

Cloning

Snowflake cloning is a core feature within the Snowflake Data Cloud, providing a highly efficient mechanism for creating virtual copies of databases, schemas, or tables. Leveraging Snowflake's unique zero-copy architecture, clones initially share underlying data with their source objects, optimizing storage costs and accelerating provisioning of development, testing, or analytics environments.

Snowflake stores data in immutable micro-partitions. Cloning operations create new metadata pointers referencing these micro-partitions instead of duplicating the raw data. Metadata updates seamlessly manage data divergence through this process, ensuring consistency and minimizing storage overhead.

Working of Snowflake clone

When you clone an object, the clone is created with the same definition as the original object. This means that the clone has the same tables, views, functions, and other objects

as the original object. The clone also has the same data as the original object at the time it was cloned.

After you clone an object, you can make changes to the clone without affecting the original object. For example, you can add, delete, or update data in the clone. You can also create new tables, views, functions, and other objects in the clone.

Access control privilege: When cloning a database or schema, the resulting clone inherits all permissions (privileges) applied to the child objects within the source. For databases, this includes schemas, tables, views, and so on; for schemas, it encompasses tables, views, and so on. However, the cloned database or schema itself does not inherit permissions granted on the original source.

In Snowflake, external stages (referencing external storage) can be cloned independently, while internal stages (within Snowflake) cannot. During database/schema cloning, external stages present in the source are replicated in the clone. However, internal table stages are cloned empty, and other internal stages are not cloned. Understanding these behaviors is important for effective data management within Snowflake.

Benefits of Snowflake cloning

Snowflake cloning feature that allows you to create an exact copy of an entire databases, individual schemas, or specific tables in Snowflake. There are many benefits to using Snowflake cloning. Here are a few examples:

- It is a fast and efficient way to create a copy of an existing Snowflake object.
- **To isolate data for a specific purpose**: You can clone a database or one or more individual tables to isolate data for a specific business use cases.
- **Development and testing**: Validate code changes, new features, or schema modifications without impacting production data.
- **Analytics and reporting**: Experiment with diverse analytical models, algorithms, and data transformations in a controlled environment.
- **Disaster recovery**: Maintain readily available backups for business continuity and regulatory compliance.

Considerations when using Snowflake cloning

There are a few things to consider when using Snowflake cloning. To maximize the benefits of cloning, organizations should:

- **Strategic clone planning**: Define clear objectives for each clone and avoid excessive cloning to streamline environment management.
- **Storage monitoring**: While initial storage overhead is minimal, monitor data divergence over time as it can gradually increase storage consumption.

- **Time travel integration**: Leverage Snowflake's Time Travel feature to clone objects at specific historical points for reproducibility, auditing, and compliance purposes.
- **Security and access control**: Implement appropriate security measures and access controls on clones to maintain data governance and confidentiality.
- **Dependencies**: Thoroughly analyze object dependencies. Cloning a view, for instance, necessitates that its referenced tables are either cloned or present in the target environment.

Overall, Snowflake cloning is a powerful and versatile feature that can be used for agile data management, accelerating development cycles, supporting diverse analytical workloads, and enhancing data protection strategies. By understanding the technical needs and adhering to best practices, organizations can leverage cloning to unlock the full potential of their Snowflake Data Cloud investments.

Replication

Snowflake's replication solution offers robust features to ensure uninterrupted access and data durability. By leveraging replication and failover/failback mechanisms, organizations can create and maintain synchronized copies of their Snowflake account objects across different regions and cloud platforms.

Snowflake offers database and share replication across all account types. However, replication of additional account objects, along with failover/failback and Client Redirect functionality, are exclusive to Business Critical (or higher) editions. Refer to the following table for high-level comparisons by Snowflake replication feature/edition matrix:

Feature	Standard	Enterprise	Business Critical	VPS
Database replication	✓	✓	✓	✓
Share replication	✓	✓	✓	✓
Replication Group	✓	✓	✓	✓
Account object (other than database/share) replication			✓	✓
Failover Group			✓	✓
Data protected with Tri-Secret Secure			✓	✓

Table 12.3: *Snowflake Replication feature/edition matrix*

The replication and failover/failback functionality consists of the following two features.

Replication and failover/failback

Snowflake's replication mechanism utilizes groups to facilitate the consistent copying of designated objects to one or more target accounts, ensuring point-in-time consistency and offering a robust failover mechanism for enhanced business continuity.

- **Replication group**: This group defines the specific objects to be replicated, the target regions or cloud platforms for replication, and the frequency of replication.
- **Failover group**: This specialized group enables both replication and the capability to seamlessly transition (failover) to the replicated objects in the event of an operational disruption.

Replicable objects encompass a range of entities, including warehouses, users, roles, databases, and shares. These objects can be strategically organized into multiple groups to align with organizational requirements.

In the event of a system failure, account replication facilitates a seamless transition of your account to an alternate region or cloud platform. Each replication and failover group maintains an independent replication schedule, allowing you to customize the frequency of replication for different groups of objects. In the case of failover, you can choose to failover all groups or selectively failover specific groups as needed.

Following are the key advantages of using Snowflake replication and failover/failback feature:

- **Granular control**: Each group can be assigned a unique replication schedule, providing flexibility to define distinct replication intervals for different sets of objects.
- **Selective failover**: In a failover scenario, administrators have the option to failover all groups collectively or to selectively failover specific groups based on operational needs.
- **Client redirect**: Automatically redirect Snowflake clients to the appropriate account, minimizing downtime and ensuring continuous operation.
- **Enhanced business continuity**: Conduct planned failovers for disaster recovery drills, ensuring preparedness and measuring recovery metrics.

Client redirect

Snowflake Client Redirect is a sophisticated feature within the Snowflake Data Cloud designed to optimize business continuity, disaster recovery, and account migration. This capability allows organizations to seamlessly redirect client connections across Snowflake accounts in different regions or cloud platforms, minimizing disruptions and ensuring uninterrupted access to critical data assets.

Client Redirect is implemented using a Snowflake connection object, a secure repository for a connection URL utilized by Snowflake clients. The connection URL's hostname incorporates organizational and connection details but does not specify the target account. Instead, an administrator designates a primary connection within a specific account, to which the connection URL dynamically directs clients.

In the event of an outage impacting the region or cloud platform hosting the primary connection, administrators can seamlessly promote a connection in an unaffected account. This ensures minimal disruption for users, who can continue using the same connection URL, with Snowflake automatically resolving it to the newly active connection.

Refer to the following operational workflow steps for how the client redirect implemented:

- **Configuration**: A designated administrator provisions a connection URL and establishes a primary connection along with linked secondary connections.
- **Client update**: Snowflake clients are configured to utilize the newly generated connection URL for establishing connections.
- **Redirection (during outage)**: In the event of a service disruption, the administrator updates the connection URL to redirect client connections to a secondary connection in an unaffected region.
- **Restoration (post outage)**: Once the outage is resolved, the administrator reverts the connection URL to direct client connections back to the original primary connection.

By leveraging Snowflake Client Redirect, organizations can proactively minimize downtime, ensure uninterrupted access to their critical data assets, and enhance their overall business resilience in the face of unforeseen disruptions. This strategic approach empowers businesses to maintain operational continuity and safeguard their data integrity even in challenging circumstances.

Business continuity and disaster recovery

Snowflake's robust business continuity and disaster recovery strategy ensures uninterrupted access to your critical data, even during major outages. This is achieved through a multi-faceted approach:

- **Data replication**: Critical account objects are replicated to a geographically separate Snowflake account, providing a real-time backup.
- **Regional outage mitigation**: In the event of an outage, the replicated data seamlessly takes over, minimizing downtime and ensuring business continuity.
- **Flexible recovery options**: Snowflake offers three distinct recovery paths:
    - **Reads before writes**: Prioritizes immediate data access, even if it's slightly outdated, ideal for short-term disruptions.

  - **Writes before reads**: Ensure all pending data transactions are processed before restoring read access, guaranteeing data integrity.
  - **Simultaneous recovery**: Strike a balance between data availability and data currency by allowing reads on potentially stale data while catching up on pending writes.
- **Seamless transition**: Snowflake's Client Redirect feature allows for automatic redirection of client connections to the new primary region, minimizing disruption for end-users.
- **Complete recovery and failback**: Once the outage is resolved, the original region is restored as the primary, and all replicated data is synchronized back, ensuring data consistency.

Snowflake's comprehensive approach to business continuity and disaster recovery guarantees that your data remains secure, accessible, and up-to-date, even in the face of unforeseen disruptions. This commitment to resilience ensures that your business operations can continue uninterrupted, fostering confidence and trust in Snowflake's platform.

Conclusion

In this chapter, we learned about the key technical features of Snowflake to protect data. Further, we also learned about the various Data Protection features like Time Travel, Fail-Safe, Data Encryption, Cloning, Replication details and how the data gets protected, benefits, etc. using these features. Hope the chapter summary helps the readers to summarize overall learning and keynotes to remember for the exam.

In the next chapter, we will learn about the interesting features of Snowflake Data Sharing and further deep dive into technical details of the data marketplace, private data exchange, shares, and how it works in real-time implementations.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **List few key features of Snowflake Data Protection:**
  - **Encryption**: Snowflake uses the industry-standard AES 256-bit encryption for robust protection. The data automatically encrypts at rest, even the cloud provider can't access the data. The data gets encrypted while moving between you and Snowflake (in transit) using TLS, a secure communication protocol.
  - **Access Control**: Snowflake's granular RBAC mechanism allows organizations to precisely control user access to specific data objects based on their roles and permissions, safeguarding sensitive data from unauthorized access.

- **Auditing**: Snowflake's comprehensive auditing capabilities enable organizations to track data access activities, including who accessed what data and when. This valuable information facilitates incident investigations and compliance with Data Protection regulations.
- **Time travel**: Snowflake's unique time travel feature empowers organizations to restore(deleted or dropped objects using UNFROP) to a previous point in time, even if it has been deleted or modified, providing a safety net against accidental data loss or corruption.
- **Fail-safe:** Snowflake fail-safe protects your data from accidental deletion or corruption for an additional seven days of data recovery by Snowflake that starts immediately after the Time Travel period ends.
- **Replication**: Snowflake proactively replicates data across multiple availability zones within a region, ensuring data availability and accessibility even if one availability zone encounters disruptions.
- **Row-level security and data masking**: Snowflake's row-level security feature allows organizations to restrict access to specific rows within a table based on user permissions, further safeguarding sensitive data such as customer PII or financial data. In addition Data masking can also help prevent the data access.
- **Disaster recovery**: Snowflake offers a range of disaster recovery options, including cross-region replication and backup to object storage, ensuring Data Protection in the event of major disasters such as natural calamities or cyber-attacks.

- **What is Time Travel in Snowflake and how this is beneficial to the organizations?**
  - Snowflake Time Travel is a remarkable feature that empowers organizations to safeguard their valuable data by providing a robust mechanism to restore data to any point in time, even if it has been deleted or modified. This feature offers a safety net against data loss incidents and ensures business continuity.
  - This remarkable capability stems from Snowflake's unique architecture, which maintains immutable snapshots of data at regular intervals. These snapshots allowing users to effortlessly retrace their data's evolution and restore it to a desired state ranging from one to 90 days.
  - The Time travel data retention supports 0 to 1 days for standard and up to 90 days for Enterprise Edition and higher.
- **What is Fail-safe Travel in Snowflake and how this is beneficial to the organizations?**

- Snowflake Fail-safe (by Snowflake) emerges as a comprehensive data recovery solution, empowering organizations to seamlessly restore their data beyond the Time Travel feature, shielding against accidental deletions, data corruption, or malicious acts.
- The Fail-safe feature is non-configurable and allows seven days of recovery of data by Snowflake that starts immediately after the Time Travel period ends.

- **What is data masking in Snowflake?**
  - Snowflake Data Masking enables organizations to mask sensitive data, such as PII, to manage and control sensitive data. This masking technique protects sensitive data from unintentional exposure while still allowing for development and testing activities.
- **What is Cloning in Snowflake and how this technique works?**
  - Snowflake cloning is a feature that allows you to create a copy of an existing Snowflake object, such as a table, schema, or database. The clone is a point-in-time copy of the source object, which means that it contains the same data as the source object at the time it was cloned.
  - Cloning operations create new metadata pointers referencing these micro-partitions instead of duplicating the raw data. Metadata updates seamlessly manage data divergence through this process, ensuring consistency and minimizing storage overhead.
- **How Snowflake Clone works?**
  - When you clone an object, the clone is created with the same definition as the original object. This means that the clone has the same tables, views, functions, and other objects as the original object.
  - After you clone an object, you can make changes to the clone without affecting the original object. For example, you can add, delete, or update data in the clone. You can also create new tables, views, functions, and other objects in the clone.
- **What is Snowflake Replication?**
  - Snowflake replication is a feature that allows you to copy data from one Snowflake database to another Snowflake database. By leveraging replication and failover/failback mechanisms, organizations can create and maintain synchronized copies of their Snowflake account objects across different regions and cloud platforms.
  - Snowflake offers database and share replication across all account types. However, replication of additional account objects, along with failover/failback and Client Redirect functionality, are exclusive to Business Critical (or higher) editions.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 13
Snowflake Data Sharing

Introduction

In this chapter, we will learn about Snowflake's secure data-sharing features, including Private data exchanges and data sharing controls to set granular permissions to control who can access your data. This chapter takes you on a journey through Snowflake's data-sharing platform, revealing how you can:

- **Access data**: Discover various ways to find and connect with the data you need.
- **Share data**: Learn how to securely share your data with others.
- **Exchange data**: Understand how data is transferred between users seamlessly.
- **See it in action**: Explore real-world examples of how the platform works.

Structure

In this chapter, we will cover the following topics:

- Account types
- Data Marketplace and data exchange
- Direct data sharing
- Private data exchanges

- Data Marketplace
- Access control options

Objectives

This chapter aims to provide comprehensive information on Snowflake Data Marketplace and further technical details about the account types, exchange, shares, and how it works in real-time implementations. By the end of this chapter, readers will gain knowledge and a clear understanding of Snowflake's data marketplace and how it empowers secure and efficient data sharing.

Account types

Snowflake offers a multifaceted account structure designed to cater to a wide range of organizational needs and data complexities. This structure encompasses several key dimensions:

Firstly, the platform offers different **Editions**, each designed with a unique set of features and capabilities. Snowflake provides the following editions along with the high level features.

- **Standard edition**: Provides core platform functionalities and is suitable for smaller organizations or those new to cloud data platforms.
- **Enterprise edition**: Offers advanced features such as materialized views, row-level security, and increased warehouse size limits, ideal for larger enterprises with complex data needs.
- **Business critical edition**: Includes additional security features like HIPAA compliance and customer-managed keys, designed for organizations with strict data governance and regulatory requirements.
- **Virtual Private Snowflake (VPS)**: Offers the highest level of isolation and security, suitable for highly sensitive data and industries with stringent compliance mandates.

Snowflake accounts can be further categorized by cost model optimizing for cost efficiency. Following are the account types by **Cost Models** providing flexibility in managing the billing structure:

- **On-Demand**: Offers a flexible, pay-as-you-go model, ideal for workloads with fluctuating demands.
- **Capacity**: Provides cost savings through pre-purchased compute resources, optimized for consistent and predictable workloads.

Snowflake accounts can also be categorized by relationship to the data or specific use cases. Following are the accounts type by Data Access Model (Data Sharing Accounts):

- **Consumer account**: The standard account type for creating and managing your own databases, tables, and data assets.
- **Reader account**: Designed for accessing and consuming data shared by Consumer Accounts. Suitable for users who primarily consume data without the need to modify underlying data.

By understanding the different account types, editions, and cost models within Snowflake is crucial for organizations to make informed decisions that align with their specific data strategies, security needs, and budgetary considerations.

Data Marketplace and data exchange

The data cloud and data exchange are another unique feature introduced by Snowflake that allows the organization to securely store and share data within or outside the organizations. This eliminates the data silos and no more complex ETL/ELT by sending the files between one or more internal or external entities.

The traditional data sharing approaches require time-consuming, costly, and laborious procedures to share even just a piece of your data from the data warehouse. To solve these critical challenges, Snowflake innovated its build for the cloud data warehouse to the idea of Snowflake data sharing.

The secure data share lets the organization share one or more selected objects in your account with other Snowflake accounts. This can be within the organization or outside accounts. This is the key unique feature of Snowflake.

Snowflake supports the following data sharing technologies to share with other Snowflake accounts:

- Direct data sharing
- Private data exchanges
- Snowflake data marketplace

Let us deep dive into each data-sharing method and how it works in the next section.

Direct data sharing

Snowflake Data Sharing is a feature that allows you to securely share live data with other Snowflake accounts or with reader accounts. This means that you can grant other people access to your data without having to copy or move it. They can then query your data directly, just as if it were their own.

Snowflake doesn't physically move your data. Instead, it creates a secure connection that allows authorized users to read-only access specific databases or tables you choose. It is like giving someone a library card to your private collection, not the books themselves.

As part of this data sharing process, Snowflake allows sharing of the following database objects:

- Databases, schemas
- Snowflake tables
- External tables
- Snowflake secured views
- Snowflake secured materialized views.
- Snowflakes Secure **User Defined Functions (UDFs)**

There are two main ways to share data with Snowflake:

- **Shares**: You can create a share, which is a pointer to a database or schema in your account. You can then share that share with other Snowflake accounts. The people with whom you share the share will be able to see and query the data in the database or schema, but they will not be able to modify it.
- **Listings**: You can create a listing, which is a way to publish your data to the Snowflake Data Marketplace. The Data Marketplace is a place where people can go to find and buy data from other Snowflake accounts. If you create a listing, people will be able to see your data and buy access to it.

Snowflake Data Sharing is a powerful process that can be used to share data with a variety of people and organizations. By establishing live, read-only connections to shared data, Snowflake Data Sharing eliminates the need for a time-consuming and resource-intensive data copy process. This real-time access ensures that all stakeholders have the most current insights, facilitating agile and informed business strategies. Moreover, Snowflake's robust security measures guarantee the protection of sensitive data throughout the sharing process.

Additional benefits of using Snowflake Data Sharing

Snowflake Data Sharing offers a robust and secure solution for real-time data collaboration. Here are some additional benefits of using Snowflake Data Sharing:

- **Enhanced security**: Granular access controls and end-to-end encryption ensure data protection at rest and in transit.
- **Scalable architecture**: Accommodates sharing of individual database objects or entire databases with a flexible number of consumers.
- **Cost optimization**: Consumption-based pricing model aligns costs with actual usage of storage and compute resources.

- **Operational efficiency**: Eliminates the need for manual data replication or transfer processes.
- **Live data access**: Provides consumers with immediate access to the latest data updates.
- **Comprehensive auditing**: Maintains detailed logs of data access and usage for compliance and governance purposes.

Additionally, reader accounts enable controlled data access for users who require read-only privileges, further enhancing security and governance. Overall, Snowflake Data Sharing empowers organizations to foster secure and efficient collaboration on live data, both internally and with external partners, while minimizing overhead costs.

Private data exchanges

Snowflake's Private Data Exchange facilitates secure and controlled data collaboration between organizations. This innovative solution empowers businesses to leverage valuable data insights while maintaining strict privacy and governance standards.

Here's how it works and operational process:

- **Data upload**: Data owners securely upload designated datasets to the exchange.
- **Access requests**: Data consumers browse available datasets and request access to specific ones.
- **Permission verification**: Snowflake verifies access permissions and ensures compliance with data privacy regulations.
- **Controlled access**: Upon approval, data consumers gain access to the data within the Snowflake platform to query and analyze the data without leaving the platform.

Following are few key benefits of using the Snowflake Private Data exchange:

- **Streamlined collaboration**: The platform streamlines the process of data sharing, fostering collaboration between organizations on data-driven projects.
- **Compliance**: Snowflake's Private Data Exchange adheres to strict data privacy regulations, ensuring compliance and mitigating risk.
- **Secure data sharing**: The platform enables organizations to securely share specific datasets with authorized parties, eliminating the need for data replication or transfer.
- **Data privacy and control**: Snowflake's stringent security measures ensure data remains confidential and accessible only to approved users, with data owners retaining full control and ownership.

Snowflake's Private Data Exchange represents a paradigm shift in data collaboration, enabling businesses to unlock the full potential of their data assets while maintaining the

highest levels of security and control. This cutting-edge solution empowers organizations to leverage data-driven insights for informed decision-making, fostering innovation and competitive advantage in the digital age.

Refer to the following figure for high-level Snowflake Private Data Exchange setup:

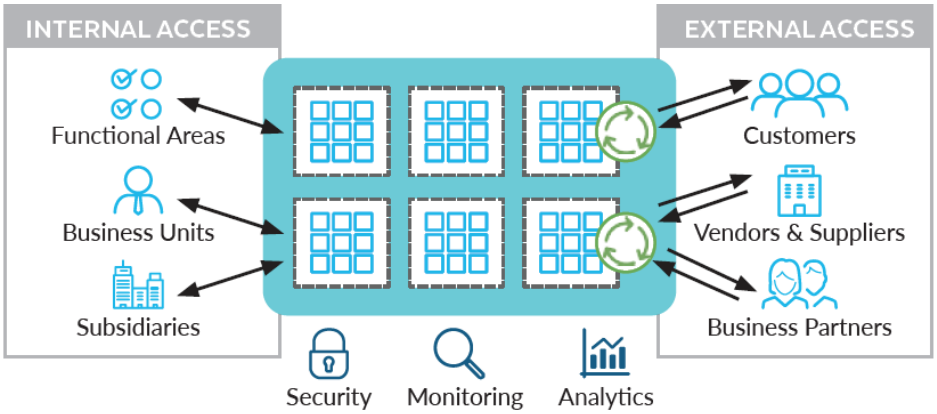

***Figure 13.1**: Snowflake Private Data Exchange*
Source: Snowflake documentation

Let us see how the Snowflake Private Data Exchange helps with the real-time scenario example:

Imagine a pharmaceutical company researching a new cancer treatment. They possess valuable clinical trial data but lack the AI expertise to analyze it effectively. Through the Private Data Exchange, they partner with a leading AI research lab. Following is the list of steps involved to setup as a data owner (Pharma Company) and access data as a data customer (research lab):

1. The pharma companies upload their anonymized clinical trial data to the exchange.
2. The research lab requests access to the data, outlining their specific research goals and demonstrating their expertise in AI-powered drug discovery.
3. Snowflake verifies credentials and grants them access to the data within its secure environment.

The research lab can now:

- Train AI models on the anonymized patient data to identify patterns and predict treatment outcomes.
- Develop a personalized treatment algorithm that factors in individual patient characteristics.
- Share their findings with the pharma company through secure visualizations and reports, without ever revealing the underlying data.

This collaboration, facilitated by the Private Data Exchange, allows the pharma company to:

- Accelerate their drug development process by leveraging the AI lab's expertise.
- Increase the success rate of their clinical trials by personalizing treatment plans.
- Ultimately bring life-saving treatments to patients faster than ever before.

This is an example of how the Snowflake Private Data Exchange empowers the organizations. It is a game-changer that fosters innovation, unlocks hidden potential, and paves the way for a brighter future fueled by data collaboration.

If you are looking to unlock the true potential of your data while safeguarding its privacy, look no further than the Snowflake Private Data Exchange. It is where collaboration meets security, and where the future of data-driven progress begins.

Data Marketplace

In today's data-driven world, organizations are constantly seeking ways to harness the power of information to gain insights, drive innovation, and achieve strategic objectives. Snowflake Marketplace emerges as a transformative platform that empowers businesses to seamlessly access, integrate, and utilize a vast array of third-party data sources, enriching their internal data landscape and fostering informed decision-making.

Enrich your data landscape with Snowflake Marketplace, it empowers you to:

- **Expand your data horizons**: Discover a diverse range of data sets, from financial market data to customer insights, to augment your internal data and gain a holistic perspective of your business and industry.
- **Accelerate time to insights**: Streamline your data access and analysis process, enabling you to extract meaningful insights swiftly and make informed decisions in real-time.
- **Optimize costs**: Eliminate the burden of maintaining your own data infrastructure and pay only for the data you consume, maximizing cost-efficiency.

Snowflake Marketplace offers a variety of data sources from the marketplace listings. Users can find the may free, free-to-try, and paid listings based on their organizations need.

Refer to the following figures for a few sample listings from the Snowflake marketplace:

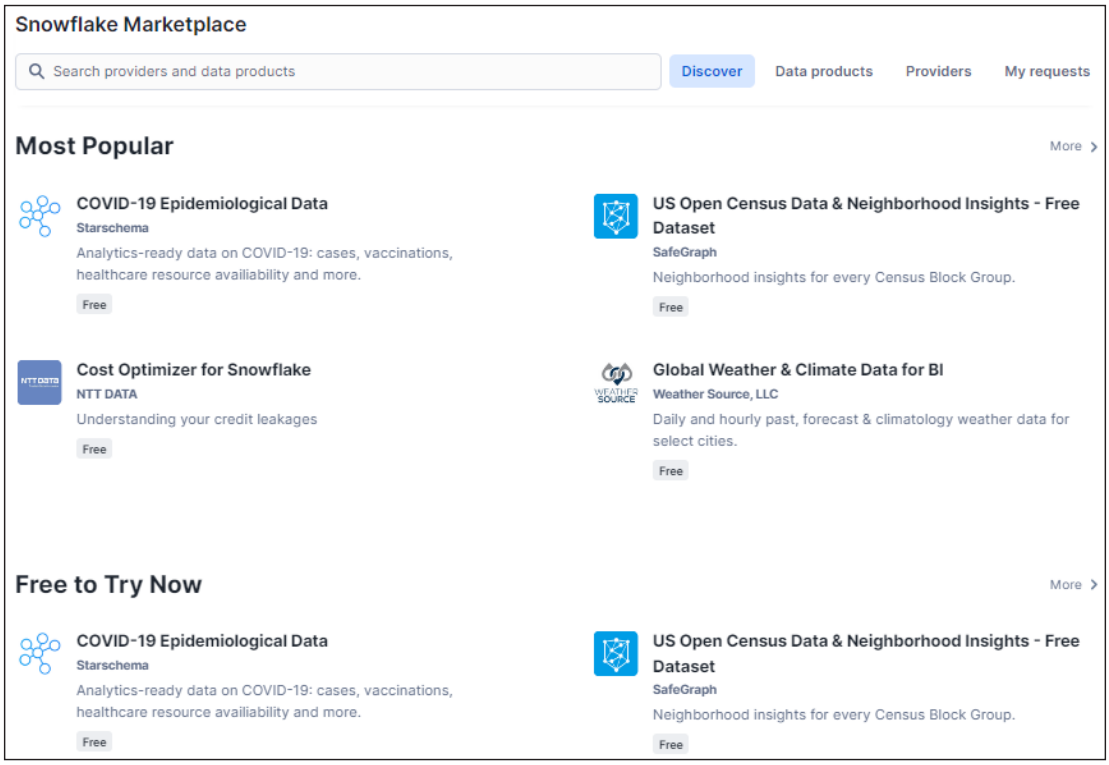

Figure 13.2: *Snowflake Marketplace Sample listings*

Uses of Snowflake Marketplace

There are a few practical reasons why you might want to use Snowflake Marketplace:

- **To get new insights into your business:** External data can help you understand your customers better, identify new market opportunities, and make better decisions.
- **To save time and effort:** Instead of spending time collecting and managing various 3rd party data, you can use Snowflake Marketplace to get access to readily available data from trusted providers.
- **To collaborate with others:** Snowflake Marketplace makes it easy to share data with other Snowflake users, so you can work together to solve problems and achieve common goals for example Geo and public reference data etc.

Data available on Snowflake Marketplace

Snowflake Marketplace offers a variety of data sources from the marketplace listings. Users can find the following free, free-to-try, and paid listings under the most popular

categories. There is a wide variety of data available on Snowflake Marketplace, following are the few example:

- **Financial data**: Stock prices, company financials, quarterly reports, and economic indicators
- **Demographic data**: Population data, income data, and education data
- **Geospatial data**: Maps, satellite imagery, and location data
- **Marketing data**: Customer data, website traffic data, and social media data

Refer to the following figures for a few Top categories listings from the Snowflake marketplace:

Figure 13.3: *Snowflake Marketplace – Top categories*

Snowflake Marketplace is a great way to get access to the data you need to make better decisions. It is easy to use, and affordable, and it has a wide variety of data available. Start using Snowflake Marketplace today.

Refer to the following figure for high-level setup of Snowflake Marketplace:

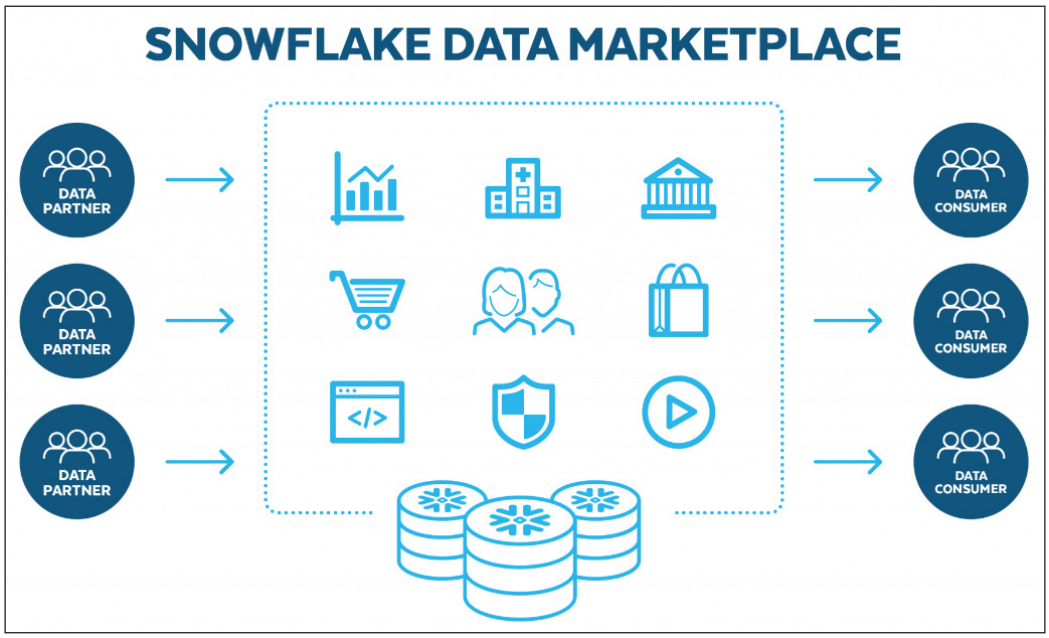

Figure 13.4: *Snowflake Marketplace*
Source: Snowflake documentation

- **Data providers** (like financial institutions, retailers, and healthcare organizations) list their data sets on the marketplace, specifying details like format, size, and cost for others to readily consume the data.
- **Data consumers** (like businesses, researchers, and analysts) can browse and search the marketplace for specific data sets they need to fuel their projects. Find the data you need faster, with clear pricing and transparent access. No more endless searches or frustrating negotiations.

As a **data provider**, you can use the listing in the marketplace to securely share the curated data sets with many Snowflake customers, not to maintain any additional steps but to maintain the data sharing with each customer; this is called market place listing.

When creating a listing, we have the option to set up, either it is publicly shared on the Snowflake marketplace or privately shared with selected customers. Marketplace listings can be free, paid, or personalized, specific to customers' tailored requirements.

The listings are further subdivided into various types to share the information and data with the customers based on the offering types:

- **Private listings**: These can be free or paid. Private listings are to share the data and information directly with another Snowflake account in any Snowflake Region.
- **Free listings**: Free listings provide access to generic data sets available in Snowflake Marketplace provide instant access to a published dataset and they are free.
- **Paid listings**: This paid listing is available privately on the Snowflake marketplace. The data provider charges consumers to access or use the data from the private listings. The paid listing is only available to consumers in a specific region and from providers in a specific region.
- **Personalized listings**: The personalized listings are the customer's specific data sets from the data providers. A personalized listing can contain tailored specific premium data to the individual customers with premium cost.

Data provider vs data consumers

What is the main difference between data provider and consumer: When sharing the data in Snowflake, the account or user that shares the data is called a **data provider**. The account or user who consumes or accepts the data is called a **data consumer**.

The following figure illustrates different types of data sharing setups in Snowflake:

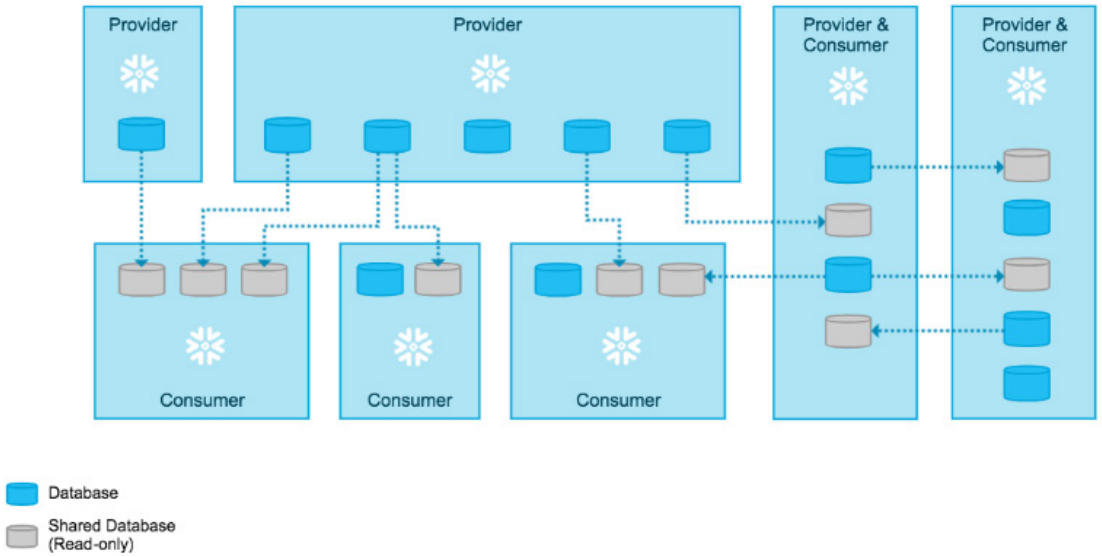

Figure 13.5: *Snowflake data provider vs consumer setup*
Source: Snowflake documentation

After getting into the Snowflake marketplace, you can do the following as a provider:

- Generate new leads and opportunities among the Snowflake customer base by free-to-use datasets listings from the marketplace.
- Securely share and integrate real-time data without creating multiple copies for each consumer.
- Eliminate time and costs to build and maintain APIs and data pipelines to deliver data to the customers.
- Secure and integrate Snowflake's access control model.
- Only pay normal storage costs for shared data.
- No limit to the number of consumer accounts with which the dataset may be shared.
- Create and publish listings that can be customized based on consumer needs.

After getting into the Snowflake marketplace, you can do the following as a consumer:

- Becoming a consumer, get instant access to explore and test as many listings.
- Access to real-time data without the need to move or transform the data
- Access to available datasets instantly and updated continually for users.
- Discover and test one or more third-party data sources.
- Query and combine shared data with your existing data or join together data from multiple data providers in Snowflake to derive new business insights.

- Receive immediate access to raw data products from vendors.
- Use the tools and technologies of your choice.
- Remove the costs of building and maintaining various data pipelines and APIs to load and update data.

The real beauty of the Marketplace lies in its democratization of data. Previously, valuable information was often siloed within companies or locked away behind expensive licenses. Now, anyone with a Snowflake account can access a vast array of data, regardless of their size or budget.

Key benefits of Snowflake Marketplace

Why Snowflake Marketplace is a big deal and what impact it made on the industry? Following are few key takeaways:

- **Unprecedented access to data**: Previously, acquiring valuable data often meant navigating complex partnerships, tedious negotiations, and hefty fees. The marketplace cuts through the red tape, offering instant access to a vast pool of data, some of which was previously unavailable.
- **Reduced time to insights**: Businesses can ditch the data-gathering legwork and focus on what truly matters: extracting insights and making data-driven decisions faster.
- **Enhanced innovation**: With a wider range of data at their disposal, businesses can explore new avenues for innovation, develop targeted marketing campaigns, and personalize customer experiences like never before.

Let us see how the Snowflake Data Marketplace helps with the real-time scenario example.

Imagine a new e-commerce start-up specializing in organic baby food. They need data to understand customer demographics, buying habits, and competitor landscapes. Instead of spending months acquiring piecemeal data from various sources, they head to the Snowflake Data Marketplace.

There, they find a treasure trove:

- Demographic data from a leading baby product retailer, revealing the age range, income levels, and geographic distribution of potential customers.
- Purchase history data from a large grocery chain, showing what types of organic baby food are popular and which brands dominate the market.
- Social media sentiment data from a company specializing in consumer insights, uncovering parent opinions and concerns about organic baby food.

By combining these data sets, the start-up gains a 360-degree view of their target audience, allowing them to:

- Fine-tune their product offerings to cater to specific customer needs and preferences.
- Craft targeted marketing campaigns that resonate with the right audience on the right platforms.
- Dynamically adjust pricing and promotions based on real-time market trends.

This is just a glimpse into the potential of the Snowflake Data Marketplace. With its ever-growing collection of data sets and its commitment to innovation, it is poised to revolutionize the way businesses access, analyze, and leverage data for success.

It is a game-changer for anyone who wants to leverage the power of data without the usual barriers. This is how the Snowflake Data Marketplace can empower businesses to make smarter decisions, unlock new opportunities, and gain a competitive edge.

Access control options

Snowflake empowers you to share data securely and precisely with its robust access control options. No more worrying about unauthorized access or accidental data leaks; you have complete control over who sees what.

Let us say we have a big box of data you want to share with others, but you only want them to see specific things. Snowflake's Data Sharing Access Control is like a set of keys and locks for that box.

- **Granularity:** It is like having many different keys, each unlocking different parts of the box. You can share an entire database, schema, table, just specific columns, or even hide certain rows based on who's looking. Data can be shared at all of these objects level and shares carry the role privileges automatically for the child objects.
- **Flexibility:** It is like being able to easily change the locks and keys. You can create different groups with different access levels, add or remove people from those groups, and even let people temporarily borrow keys for specific tasks. This means you can share data securely and efficiently, giving people exactly what they need without compromising sensitive information.

Here is a breakdown list of the access control key features:

- Granular privileges:
    - **Object-leve**l: Share specific databases, schemas, tables, or even columns within a table. Need to share only customer names and email addresses? No problem!
    - **Privilege-level**: Grant different levels of access, from read-only to full **create, read, update, and delete** permissions. Imagine letting a marketing team see sales data but not edit it.

Example: A healthcare organization shares a database with researchers. They grant read-only access to patient demographics and diagnoses but mask sensitive information like social security numbers.

- Secure views and materialized views:
    - **Data masking**: Hide sensitive data with dynamic masking policies. Think of blurring faces in photos or replacing real names with placeholders.
    - **Row-level security**: Filter data based on user attributes or custom criteria. Imagine researchers only seeing data from specific geographic regions.

      Example: A financial services company shares customer transaction data with regulators. They use a secure view to mask account balances exceeding a certain threshold.
- **Secure User-Defined Functions(UDFs):**
    - Data transformation: Apply custom logic before user's access data. Think anonymizing email addresses or converting currencies.
    - Data redaction: Permanently remove sensitive information before sharing. Imagine removing credit card numbers before sharing sales data.

      Example: A retail chain shares product sales data with suppliers. They use a secure UDF to remove store location identifiers before sharing.
- **Granting access:**
    - **Roles:** Assign privileges to roles, and then grant roles to users or other roles. Makes managing access for large groups easier.
    - **Direct Grants:** Grant privileges directly to users or shares for finer-grained control. Useful for individual users or external partners.

      Example: A research institute shares a climate data set. They create a *climate scientist* role with read-only access and assign it to researchers. They also grant direct read-only access to a specific environmental agency.
- **Data sharing options:**
    - **Listings**: Share data publicly or privately on the Snowflake Data Marketplace.
    - **Direct Shares**: Share data directly with specific accounts.
    - **Data Exchanges**: Manage and control data sharing within a group of accounts.
- **Tag-Based Access Control:**
    - **Tag your data**: Snowflake allows users to label tables, columns, or even databases with specific tags based on their sensitivity or purpose like *Finance*, *PII*, *Marketing*, and so on.

- **Set access rules**: Based on these tags, define rules that grant access to specific users or roles. Let's say, *Users with the 'Finance' tag can see 'Finance' data, but those with 'Marketing' can't*.
- **Automatic enforcement**: Snowflake automatically grants or denies access based on user tags and data tags, like a smart librarian checking your tag before handing you a book. This is very flexible way of managing the access rules for various scenarios.

Conclusion

In this chapter, we learned about the Snowflake Data Marketplace and further technical details about the account types, data marketplace, exchange, and how it works in real-time implementations. In addition, we also learned about Snowflake's private data exchange and shares, and how the access control options are implemented to manage and control the data securely in the data marketplace.

The Snowflake platform keeps evolving with many new features and latest additions. In the next chapter, we will learn about few latest additions to the data cloud, such as Snowpark, dynamic and iceberg tables, and so on.

Points to remember

Refer to the following summary for a quick reference to what we learned so far in this chapter:

- **List various types of data sharing methods available in Snowflake:**

  Snowflake supports the following data sharing technologies to share data with other Snowflake accounts:

  - Direct data sharing
  - Data or private exchanges
  - Snowflake data marketplace

- **What is Snowflake Shares and how this works?**

  Snowflake Data Sharing is a feature that allows you to securely share live data with other Snowflake accounts or with reader accounts. In Snowflake, organizations can grant other people access to the data without having to copy or move it. They can then query your data directly, just as if it were their own.

  Here is how it works. There are two main ways to share data with Snowflake:

  - **Shares**: You can create a share, which is a pointer to a database or schema in your account. You can then share that share with other Snowflake accounts.

The people with whom you share the data will be able to see and query the data in the database or schema, but they will not be able to modify it.

- **Listings**: You can create a listing, which is a way to publish your data to the Snowflake Data Marketplace. The Data Marketplace is a place where people can go to find and buy data from other Snowflake accounts. If you create a listing, people will be able to see your data and buy access to it.

- **What is Private data exchange and how this works?**

  Imagine a world where businesses can share valuable data insights without fearing exposure or compromising privacy. That is the magic of Snowflake's Private Data Exchange, a secure platform that lets organizations collaborate on data-driven projects without ever relinquishing control of their sensitive information.

  Here is how it works:

  - Data owners upload their private data sets to the exchange.
  - Data consumers express interest in specific data sets and request access. Snowflake acts as a trusted intermediary, verifying permissions and ensuring compliance with data privacy regulations.
  - Once approved, data consumers can query and analyze the data within the Snowflake platform, but they never actually download or possess the underlying data. This way, data owners retain complete ownership and control.

- **What is a Snowflake Marketplace and key benefit of using the Marketplace?**

  Snowflake Marketplace is a great way to get access to the data you need to make better decisions. It is easy to use, and it has a wide variety of data available.

  Key benefits of using Snowflake Marketplace:

  - Browse the shelves: Search through thousands of data sets by category, keyword, or industry.
  - Try before you buy: Most providers offer free samples so you can test the quality before committing.
  - Seamless integration: Snowflake connects directly to your existing data warehouse, so you can start analyzing right away.

- **Data Marketplace: What is the difference between Data Providers and Data Consumers?**

  - **Data provider**: When sharing the data in Snowflake, the Account or user that shares the data is called a data provider.
  - **Data consumer**: The account or user who consumes or accepts the data is called a data consumer.

Chapter 14
Snowflake Latest Additions

Introduction

This chapter delves into Snowflake's latest advancements: Snowpark, Dynamic tables, Iceberg tables, and the Snowflake Cortex. Snowpark is a revolutionary engine that executes code directly within the Snowflake Data Cloud, enabling streamlined data processing and eliminating the need for moving data around. Dynamic tables offer unmatched flexibility, allowing you to define schemas on the fly and adapt to evolving data structures. Iceberg tables, meanwhile, enhance data governance with their robust partitioning and data quality features. Snowflake Cortex is a service that seamlessly integrates **Large Language Models (LLMs)** into your Snowflake environment, eliminating the need for complex external configurations. We will explore how these innovations are continuously evolving to empower the Snowflake community and unlock the full potential of the platform for data management and analytics.

Structure

In this chapter, we will cover the following topics:

- Snowflake Snowpark
- Snowflake Dynamic Tables
- Snowflake Iceberg table
- Snowflake Cortex

Objectives

This chapter provides a comprehensive exploration of Snowpark, Dynamic, Iceberg tables and the latest Snowflake Cortex. By mastering these cutting-edge features, you will unlock the full potential of Snowflake for all your data management and analytics needs. This chapter serves as your guide to becoming a highly proficient Snowflake user, empowering you to extract maximum value from the platform.

Snowflake Snowpark

Harness data's potential with unprecedented agility: Snowflake Snowpark bridges the gap between data and development, empowering you to create, collaborate, and innovate directly within Snowflake's data cloud. This groundbreaking framework seamlessly integrates familiar programming languages like Python, Java, and Scala into Snowflake's powerful ecosystem, unlocking a new era of data-driven decision-making.

Key features of Snowpark

The following are the key features of Snowpark:

- **Unified development experience:**
    - Snowflake Snowpark's visual interface with code editor and data visualization panels write, execute, and manage code within Snowflake's secure and scalable environment.
    - Leverage familiar DataFrame **Application Programming Interfaces** (**APIs**) for intuitive data exploration and transformation.
- **Native machine learning integration:**
    - Train and deploy machine learning models directly on Snowflake's data.
    - Simplify model development and deployment cycles for faster insights.
- **Streamlined data pipelines:**
    - Build efficient data processing workflows within Snowflake, eliminating the need for separate tools and infrastructure.
    - Enhance productivity and reduce data transfer costs.
- **Collaborative environment:**
    - Share code and collaborate seamlessly with your team.
    - Foster knowledge sharing and accelerate innovation.
    - Unlock the full potential of your data

Examples of Snowpark in action

In this section, we will discuss the examples of Snowpark in action:

- **Marketing analytics**: Analyze customer behavior, segment audiences, and personalize campaigns using Python's rich analytics libraries directly within Snowflake.
- **Fraud detection**: Develop machine learning models in Scala to detect anomalies in financial transactions, protecting against fraud.
- **Supply chain optimization**: Create Java-based data pipelines to streamline inventory management, track logistics, and optimize costs.

Key benefits of Snowpark

The following are the key benefits of Snowpark:

- **Enhanced productivity**: Work in your preferred language, streamline workflows, and eliminate data movement overhead.
- **Accelerated time to insights**: Rapidly transform, model, and analyze data within Snowflake's scalable platform.
- **Seamless collaboration**: Foster a connected data community, empowering knowledge sharing and faster results.
- **Unleashed innovation**: Explore, experiment, and build cutting-edge data solutions directly within Snowflake's environment.

Snowpark is a game-changer for developers and data teams alike. Embrace its power to drive innovation, accelerate insights, and transform data into unparalleled value.

Snowflake Dynamic Tables

Snowflake's dynamic tables represent a paradigm shift in data pipeline development. These innovative tables leverage SQL queries to automatically materialize and maintain the query's results. This streamlines data processing by eliminating the requirement for dedicated target tables and custom transformation logic. Dynamic tables seamlessly integrate data joins and aggregations across diverse source objects.

Imagine a table that magically stays up-to-date whenever the data it's based on changes. That's basically a Dynamic table in Snowflake. Instead of writing complex code to update tables, you define a Dynamic table with a simple SQL query that tells Snowflake what data you want. Snowflake then automatically fetches the data, processes it based on your query, and keeps the table updated.

- Key use cases:
    - Real-time dashboards and analytics

- Personalization and recommendation engines
- Fraud detection and anomaly monitoring
- Log analysis and troubleshooting

Key features of dynamic tables

The following are the key features of Dynamic Tables:

- **Simplified data pipelines**: Create and manage data pipelines with minimal SQL code, reducing development complexity and time-to-insights.
- **Self-maintaining:** Managed by Snowflake, an automated refresh process updates the dynamic tables with the changes made to the base objects.
- **Adaptable schema**: They can handle changes in the structure of your data sources, eliminating manual schema updates.
- **Optimized for streaming**: They excel at handling real-time or frequently updated data, making them ideal for sensor readings, clickstream data, stock prices, and more.
- **Continuous data ingestion**: Seamlessly ingest and process streaming data in real-time, keeping your tables fresh and relevant.
- **Optimized partitioning**: Snowflake intelligently partitions your data behind the scenes, ensuring efficient query performance and cost management.

Here is an example of how to create and query the Dynamic table in Snowflake:

```
SQL:
-- Sample Syntax/Code to Create Dynamic table based on the underlying sales
data
CREATE OR REPLACE DYNAMIC TABLE sales_summary (region, total_sales)
  TARGET_LAG = '10 minutes'
-- The target lag time is 10 minutes, which means that the data in the
dynamic table should ideally be no more than 10 minutes older than the data
in sales_table
    WAREHOUSE = compute_wh
    AS
     SELECT region, sum(sales) as total_sales
       FROM sales_table
       Group by region;
-- Sample Syntax/Code to query the dynamic table
Select * from sales_summary;
```

Examples of dynamic tables

Refer below for a few examples of how the Dynamic tables can be implemented in real time scenarios:

- **Real-time sales dashboard:**
    - Create a Dynamic Table to continuously ingest sales data from multiple sources (for example, point-of-sale systems, online transactions).
    - Build a dashboard that reflects real-time sales performance, inventory levels, and customer behavior, enabling immediate insights and actions.
- **Personalized product recommendations:**
    - Capture user interactions and preferences in a Dynamic Table.
    - Use this data to generate tailored product recommendations in real-time, enhancing customer experiences and driving sales.
- **Fraud detection:**
    - Analyze streaming financial transactions in a Dynamic Table.
    - Implement machine learning models to identify suspicious patterns and trigger alerts, protecting your business from fraudulent activity.
- **Log analysis:**
    - Collect application logs in a Dynamic Table.
    - Monitor system health, troubleshoot issues, and identify performance bottlenecks in near real-time, ensuring smooth operations.

Benefits of Dynamic Tables

The following are the benefits of Dynamic Tables:

- **Accelerated time to insights**: Get faster answers from your data with always-fresh tables.
- **Reduced development complexity**: Build data pipelines with simpler SQL, saving time and effort.
- **Optimized performance and cost**: Snowflake's intelligent partitioning and adaptive query optimization ensure efficient resource usage.
- **Reduced complexity**: Streamline data pipelines and eliminate the need for manual refreshes or complex **change data capture** (**CDC**) processes.
- **Enhanced data freshness**: Stay ahead of the curve with real-time insights from streaming data.
- **Cost optimization**: Pay only for the data you need and avoid over provisioning resources.

- **Enhanced data quality**: Ensure consistency and accuracy with automated updates and error handling.

Dynamic Tables are a powerful addition to Snowflake's data platform, empowering organizations to embrace the dynamic nature of data and unlock unprecedented agility and insights.

Snowflake dynamic tables vs. materialized views

Snowflake offers these innovative table types to address diverse data storage and management needs. By understanding their unique characteristics and ideal use cases, businesses can make informed decisions to align their data strategies with their specific goals.

Snowflake offers two primary techniques for materializing data to enhance query performance: materialized views and dynamic tables. Understanding the strengths and use cases of each approach is crucial for optimizing data access within your Snowflake environment.

- **Materialized views (MVs):**
    - Pre-compute query results based on a single base table, facilitating significantly faster retrieval for subsequent queries, especially those involving complex aggregations, projections, or selections.
    - Ideal for frequently executed queries on large datasets where real-time data isn't essential.
    - Data in materialized views remains current due to automatic refresh mechanisms triggered by updates in the underlying base table.
- **Dynamic tables:**
    - Materialize the results of a complex query, potentially involving joins and unions across multiple base tables. This enables the creation of multi-level data pipelines for data transformation.
    - Well-suited for scenarios requiring complex data transformations before querying or building data pipelines.
    - Data freshness depends on the refresh schedule; queries leverage the latest data within the specified target lag time.

By understanding the strengths and limitations of each technique, you can effectively leverage Snowflake's capabilities to optimize query performance and build robust data pipelines by choosing the right approach.

- **Key considerations:**
    - **Query complexity**: Materialized views are limited to single base tables and simpler queries. Dynamic tables excel with complex queries and multi-table joins.

- **Data freshness**: Materialized views offer real-time data reflection. Dynamic tables provide data with a specific lag based on refresh schedules.
- **Automation**: Materialized views benefit from automatic refresh functionality. Dynamic tables require explicit refresh scheduling.

Snowflake Iceberg Table

Snowflake Expands Data Management Options with Apache Iceberg Integration. This new innovative offering seamlessly integrates Snowflake's robust platform management and exceptional performance with the open-source Apache Iceberg format for external data storage. This empowers organizations to leverage Snowflake's strengths while maintaining control over their data in a widely adopted open format.

Think of Iceberg tables like a well-organized library. Each data file is like a book, labeled with its contents and organized within designated shelves (partitions). A detailed catalog (metadata) tells you which books are available, their versions, and where to find them. This makes finding specific information much faster and easier. That is the essence of Snowflake Iceberg tables, bringing order and efficiency to your data storage.

Snowflake Iceberg tables area special type of table format within Snowflake. They offer three key differences from Snowflake's native format:

- **Open format**: Iceberg uses an open-source format, meaning it is not tied to Snowflake's proprietary system. This gives you flexibility and interoperability with other tools and platforms.
- **External storage**: Iceberg tables store their data and metadata in your chosen external cloud storage location (for example, AWS S3, Azure Blob Storage or Google Cloud Storage), keeping you in control of your data location. Please note that Snowflake does not manage or maintain these external storage locations; therefore, users are solely responsible for all aspects of data management, including configuration of data protection and recovery procedures. Snowflake does not provide any fail-safe storage mechanisms for Iceberg tables.
- **Snapshot-based**: Iceberg captures the state of your data at specific points in time (snapshots), allowing you to easily roll back to previous versions or query historical data.

Key features of Iceberg table

The following are the key features of the Iceberg table:

- **Open storage**: Store your data in your preferred cloud storage (for example, AWS S3, Azure Blob Storage) instead of Snowflake's native storage, giving you more control and potentially lower costs.

- **Acid transactions**: Perform reliable data updates with guaranteed atomicity, consistency, isolation, and durability, ensuring data integrity even in case of failures.
- **Schema evolution**: Easily adjust your table schema over time without downtime or data loss, adapting to changing data structures.
- **Time travel and snapshots**: View historical versions of your data at any point in time, enabling data analysis and rollback to previous states.
- **Partitioning and compaction**: Optimize query performance and storage efficiency by organizing and filtering data based on specific criteria.

Iceberg table billing model

Snowflake's billing model for Iceberg tables encompasses charges for virtual warehouse (compute) usage and cloud services incurred during table operations. However, it's important to note that Snowflake does not bill for the following:

- **Iceberg table storage costs**: These are billed directly by your cloud storage provider.
- **Active bytes used by Iceberg tables**: While not billed, Snowflake provides visibility into the storage occupied by these tables through the **`INFORMATION_SCHEMA.TABLE_STORAGE_METRICS`** and **`ACCOUNT_USAGE.TABLE_STORAGE_METRICS`** views.

Important consideration: If your Snowflake account and external volume reside in different regions, egress charges may apply from your cloud storage provider when querying the table.

Example of Iceberg table

Imagine you have a huge dataset of website traffic logs stored in Iceberg tables on Azure Blob Storage. You can analyze these logs using SQL queries within Snowflake, just like you would with native Snowflake tables. However, you can also use other tools that support Iceberg, like Python scripts, to access and manipulate the data directly in your cloud storage.

Here are examples of how to create Snowflake Iceberg table. Iceberg Tables can either use Snowflake, AWS Glue, or object storage as the catalog.

Example 1: Creating an Iceberg Table with Snowflake as the Catalog

This example defines an Iceberg table named **`customer_iceberg`** with columns for **`customer_id`**, **`name`**, and **`city`**. Snowflake will manage the Iceberg catalog and store data in its internal storage.

```
SQL:
-- Sample Syntax/Code for creating iceberg table using Snowflake Catalog
CREATE OR REPLACE ICEBERG TABLE customer_iceberg
(
customer_id INT NOT NULL,
  name STRING,
  city STRING
)
  CATALOG = 'SNOWFLAKE'
  EXTERNAL_VOLUME = 'my_external_volume'
  BASE_LOCATION = 'my/relative/path/from/external_volume';
```

Example 2: Creating an Iceberg Table with AWS Glue as the Catalog

This example creates an Iceberg table named **`inventory_iceberg`** that uses the AWS Glue Data Catalog.

```
SQL:
-- Sample Syntax/Code for creating iceberg table using AWS Glue as the
Catalog
CREATE ICEBERG TABLE inventory_iceberg
  EXTERNAL_VOLUME='glue_catalog_volume'
  CATALOG='glue_catalog_integration'
  CATALOG_TABLE_NAME='my_glue_catalog_table'
  CATALOG_NAMESPACE='icebergcatalogdb2';
```

Example 3: Creating an Iceberg Table that uses object storage

This example defines an Iceberg table named **`sales_iceberg`** from iceberg metadata in external cloud storage.

```
SQL:
-- Sample Syntax/Code for creating iceberg table using object storage
CREATE ICEBERG TABLE sales_iceberg
  EXTERNAL_VOLUME='icebergMetadataVolume'
  CATALOG='icebergCatalogInt'
  METADATA_FILE_PATH='path/to/metadata/v1.metadata.json';
```

Key benefits of Iceberg Table

In essence, Iceberg Tables combine the familiar structure and power of Snowflake tables with the open-source Iceberg format for data storage. Following are the key benefits of Iceberg table.

- **Cost optimization**: Store data in your preferred cloud storage, potentially reducing costs compared to Snowflake's native format.
- **Interoperability:** Use Iceberg tables with various tools and platforms, increasing flexibility and avoiding vendor lock-in.
- **Data versioning**: Easily roll back to previous versions or query historical data, safeguarding against accidental changes or providing insights into past trends.
- **Improved performance**: Efficient data organization and querying capabilities lead to faster data access and analysis.

In essence, Snowflake Iceberg tables offer a powerful combination of open-source flexibility, cloud storage control, and version control, making them ideal for organizations that demand efficient data management and interoperable solutions.

Iceberg table limitations

Following are few limitations currently not supported by Iceberg tables. Snowflake is constantly working to improve Iceberg table support, so these limitations may be addressed in future releases.

- Iceberg tables with external volume are only supported in the same cloud and region as the Snowflake account. Cross-region and Cross-cloud Iceberg tables are not supported.
- Only parquet format is supported for data files.
- You can only create permanent Iceberg tables; transient or temporary Iceberg tables are not supported.
- Time travel in Spark is not supported for Snowflake-managed Iceberg tables.
- You cannot create clones or replicate Snowflake Iceberg tables.
- SnowGov regions – Create and working with Iceberg tables.

Snowflake Cortex

Snowflake Cortex - Democratizing LLM Access: Imagine a world where you can leverage the power of LLMs directly within your data workflows. Snowflake Cortex makes this a reality.

Snowflake Cortex acts as a bridge, seamlessly connecting you to the world of LLMs. It eliminates the complexities of managing and integrating these powerful AI models into your data workflows. With Cortex, you can leverage LLMs directly within your SQL queries and Python code, opening doors to exciting new possibilities for data analysis and application development. This is the brand new latest feature in Snowflake, so user will continue to see more functionalities as part of the production improvement to Cortex.

Key features of Snowflake Cortex

The following are the key features of Snowflake Cortex:

- **Effortless LLM Integration**: Snowflake Cortex offers an intuitive **user interface** (**UI**) that makes interacting with LLMs as simple as point-and-click. No coding experience is necessary! Additionally, for more technical users, Cortex allows incorporating LLM functionalities directly into SQL queries and Python code.
- **Security at the Forefront**: Security is paramount when dealing with data. Snowflake's robust security framework extends to LLM usage within Cortex. You can be confident that your data remains protected throughout the analysis process.
- **A Marketplace of Choice**: Snowflake Cortex offers access to a variety of LLM models through the Snowflake Marketplace. You can choose from specialized models trained for specific tasks or general-purpose models for broader applications.

Benefits of using Snowflake Cortex

The following are the key Benefits of Snowflake Cortex:

- **Enhanced data exploration:** LLMs can unlock hidden insights from your data. Use them to generate summaries of complex datasets, identify trends and patterns, or even translate languages within your data for a more comprehensive global view.
- **Supercharge data applications:** Incorporate LLMs into your data applications to create more interactive and insightful experiences for users. Imagine a data dashboard that can answer user questions in natural language or generate custom reports based on specific requests.
- **Streamlined data workflows:** Snowflake Cortex simplifies the process of incorporating LLMs into your data pipelines. This reduces development time and allows you to focus on extracting value from your data.
- **Democratize data science:** Cortex empowers both data scientists and business users to leverage the power of LLMs. Data scientists can use Cortex to build sophisticated models, while business users can gain valuable insights from data through intuitive LLM interactions.

The overall conclusion, Snowflake Cortex is a game-changer for businesses looking to leverage the power of LLMs for data analysis. Snowflake Cortex removes the barriers to entry for using LLMs in your data Cloud. It provides a secure, user-friendly, and feature-rich environment to unlock the potential of these transformative AI models. With Snowflake Cortex, you can transform your data analysis processes and develop innovative LLM-powered applications, all within the trusted Snowflake Data Cloud.

Conclusion

In this chapter, we learned comprehensive exploration of Snowpark, Dynamic, Iceberg tables and the latest Snowflake Cortex. By mastering these cutting-edge features, you unlocked the full potential of Snowflake for all your data management and analytics needs and empowered you to extract maximum value from the platform.

The next chapter is going to be the model questions that will help the readers practice what they expect in the SnowPro Exam. This will help the individuals to analyze their performance and better prepared for the exam.

This book serves as your primary resource as you start the Snowflake journey. By providing a solid foundation in Snowflake's architecture and functionalities, this guide empowers you to confidently navigate complex data warehousing tasks and projects. Ultimately, this proficiency can help readers achieve the SnowPro certification, a valuable validation of your expertise in this in-demand technology.

Points to remember

Refer to below summary for a quick reference to what we learned so far in this chapter:

- **What is Snowpark and the benefits of using it?**

  Snowpark represents a paradigm shift in data processing within the Snowflake Data Cloud. It empowers developers to securely deploy and execute code written in Python, Java, or Scala directly on the Snowflake platform. This eliminates the need for data movement, fostering streamlined workflows and enhanced performance.

  Snowpark offers a comprehensive suite of libraries that provide a familiar DataFrame API and native machine learning functionalities. This enables the construction of scalable data pipelines, machine learning workflows, and data applications entirely within the secure confines of Snowflake.

  Following are the key benefits of using the Snowpark. By leveraging Snowpark, organizations can benefit from:

  - **Simplified development:** Utilize familiar programming languages, streamlining the development process for data engineers and data scientists.
  - **Unparalleled performance:** Process data directly within Snowflake's elastic data warehouse, achieving exceptional query execution speeds.
  - **Enhanced security:** Maintain centralized governance and control over data and code within the Snowflake environment.
  - **Cost optimization:** Eliminate the need for additional data movement infrastructure, potentially reducing overall costs.

- **Describe Snowflake Dynamic Table and its key benefits:**

Snowflake Dynamic Tables are a new way to build data pipelines. Instead of writing complex code to manage data pipelines, you define a simple SQL query that transforms data from other tables. Snowflake takes care of the rest, refreshing the table with new information and keeping it current.

Key benefits of dynamic tables:

- **Simplified data pipelines:** Build data pipelines with easy-to-understand SQL statements, ditching complex coding.
- **Automatic updates:** No need to schedule refreshes manually, the table updates itself as the source data changes.
- **Focus on data transformation:** Spend less time on managing pipelines and more time on transforming data for analysis.
- **Efficient updates:** Dynamic Tables only update the new or changed information, making refreshes faster.

- **Snowflake Dynamic Tables Vs Materialized Views:**
  - **Materialized Views (MVs)**: Pre-computed results for fast querying, ideal for large datasets and frequent, simple queries. Data stays current with automatic refreshes. Best fit for Simple queries on large datasets needing fast response (real-time not critical).
  - **Dynamic Tables**: Materialized results of complex queries (joins, unions). Great for data pipelines and transformations. Data has a lag based on refresh schedule. Best fit for Complex data pipelines or queries with acceptable lag.
- **Describe Snowflake Iceberg Table and its key benefits:**

  Imagine a Snowflake table that stores data in your own cloud storage (like Amazon S3 or Google Cloud Storage) instead of Snowflake's internal storage. That's basically an Iceberg table. Snowflake Iceberg tables combine the familiar feel and performance of Snowflake tables with the flexibility of open-source formats like Apache Iceberg and Parquet.

  Key benefits of using Iceberg tables with Snowflake include:

  - **Cost-efficiency**: You only pay for the storage you use in your cloud provider, not Snowflake's storage charges.
  - **Openness**: Leverages open formats like Apache Iceberg and Parquet, making your data future-proof and usable across different systems.
  - **Performance**: Snowflake can optimize data access for reads and updates, keeping queries fast.
  - **Control**: You have more control over your data's location and lifecycle management.

Overall, Snowflake Iceberg tables are a powerful option for handling large datasets in Snowflake while keeping costs down and maintaining control over your data.

- **Snowflake Cortex**: Snowflake Cortex is a service that adds artificial intelligence and machine learning capabilities to Snowflake, a data cloud platform. In simple terms, Snowflake Cortex is a service that adds superpowers to your Snowflake data warehouse. It does this with two main tools:
    - **Large language models (LLMs)**: These are AI models that can understand and process human language. Snowflake Cortex lets you use these models to analyze text data, translate languages, write different kinds of creative text formats, and more, all directly within Snowflake.
    - **Machine learning (ML) functions**: These are tools that can find patterns in your data and use those patterns to make predictions. With Snowflake Cortex, you can use these functions to analyze your data and get insights without needing to be an ML expert.
- **What are the key benefits of using Snowflake Cortex?**

  Overall, Snowflake Cortex can help you get more value out of your data by making it easier to use AI and ML tools. Following are few key benefits using Snowflake Cortex:
    - **Ease of use:** You can leverage powerful AI features without needing to set up or manage complex AI infrastructure.
    - **Security:** Because everything runs within Snowflake, your data stays secure.
    - **Cost-effectiveness:** Snowflake Cortex is optimized to run efficiently.
    - **Speed:** You can get AI-powered insights from your data quickly.

Chapter 15
Snowflake Knowledge Test

Introduction

In this chapter, we have four sets of model questions with answer keys to help the readers practice what they expect in the SnowPro Exam. Each model test consists of 25 questions to evaluate learner knowledge across different domains. This will help the individuals to analyze their performance and prepare for the exam using these sample questions.

Structure

Topics to be covered: This chapter is going to cover the following topics:

- Model Test 1
- Model Test 2
- Model Test 3
- Model Test 4
- Answer key

Objectives

This chapter evaluates the reader's knowledge across different domains for the SnowPro exam. This helps the individuals what they can expect in the SnowPro Exam to analyze their performance and be better prepared for the exam using these knowledge tests.

SnowPro: Model test 1

1. **What are the three layers that make up Snowflake's architecture? (Choose three.)**
    a. Compute
    b. Tri-Secret Secure
    c. Storage
    d. Cloud Services
2. **In which layer of its architecture does Snowflake store its metadata statistics?**
    a. Cloud Services Layer
    b. Compute Layer
    c. Storage Layer
    d. Database Layer
3. **True or False: Snowflake's data warehouse was built from the ground up for the cloud.**
    a. False
    b. True
4. **What is the minimum Snowflake edition that customers planning on storing protected information in Snowflake should consider for regulatory compliance?**
    a. Standard
    b. Premier
    c. Enterprise
    d. Business Critical Edition
5. **True or False: A table in Snowflake can only be queried using the Virtual Warehouse that was used to load the data.**
    a. True
    b. False
6. **Snowflake provides a mechanism for its customers to override its natural clustering algorithms. This method is:**
    a. Micro-partitions
    b. Clustered partitions
    c. Key partitions
    d. Clustering keys
7. **True or False: A single database can exist in more than one Snowflake account.**
    a. False
    b. True

8. **Select the different types of Internal Stages: (Choose three.)**
    a. Table Stage
    b. User Stage
    c. Named Stage
    d. Schema Stage
9. **Credit Consumption by the Compute Layer (Virtual Warehouses) is based on: (Choose two.)**
    a. Warehouse size
    b. Number of clusters for the Warehouse
    c. Number of users
    d. Amount of data processed.
10. **Which of the following commands sets the Virtual Warehouse for a session?**
    a. USE WAREHOUSE <<warehouse name>>
    b. COPY WAREHOUSE FROM <<config file>>
    c. SET WAREHOUSE = <<warehouse name>>
    d. USE VIRTUAL_WAREHOUSE <<warehouse name>>
11. **Which object allows you to limit the number of credits consumed within a Snowflake account?**
    a. Resource Monitor
    b. Account Usage Tracking
    c. Warehouse Limit Parameter
    d. Credit Consumption Tracker
12. **Which of the following objects can be cloned? (Choose four.)**
    a. Named File Formats
    b. Tables
    c. Schemas
    d. Shares
    e. Users
    f. Databases
13. **Which of the following connectors allows Multi-Factor Authentication (MFA) authorization when connecting? (Choose all that apply.)**
    a. ODBC
    b. JDBC
    c. Python

d. SnowSQL

e. Snowflake Web Interface (UI)

14. **True or False: It is possible to unload structured data to semi-structured formats such as JSON and Parquet.**

    a. True

    b. False

15. **Which of the following statements are true of Virtual Warehouses? (Choose all that apply.)**

    a. Customers can change the size of the Warehouse after creation.

    b. A Warehouse can be resized while running.

    c. A Warehouse can be configured to suspend after a period of inactivity.

    d. A Warehouse can be configured to auto-resume when new queries are submitted.

16. **Select the three types of tables that exist within Snowflake. (Choose three.)**

    a. Temporary

    b. Transient

    c. Provisional

    d. Permanent

17. **True or False: Query IDs are unique across all Snowflake deployments and can be used in communication with Snowflake Support to help troubleshoot issues.**

    a. True

    b. False

18. **Snowflake provides two mechanisms to reduce data storage costs for short-lived tables. These mechanisms are: (Choose two.)**

    a. Temporary Tables

    b. Transient Tables

    c. Provisional Tables

    d. Permanent Tables

19. **What is the recommended Snowflake data type to store semi-structured data like JSON?**

    a. VARCHAR

    b. RAW

    c. BLOB

    d. VARIANT

20. **Which of the following are common use cases for zero-copy cloning? (Choose three.)**
    a. Quick provisioning of Dev and Test/QA environments
    b. Data backups
    c. Point-in-time snapshots.
    d. Performance optimization
21. **True or False: When a data share is established between a Data Provider and a Data Consumer, the Data Consumer can extend that data share to other Data Consumers.**
    a. True
    b. False
22. **If a Small Warehouse is made up of 2 servers/clusters, how many servers/clusters make up a Medium Warehouse?**
    a. 4
    b. 16
    c. 32
    d. 28
23. **Virtual Warehouses are charged in credits which of the following are true:**
    a. The warehouse is charged on a per-minute basis with a minimum 60 seconds
    b. The warehouse is charged on a per-second basis with a minimum 60 seconds
    c. The warehouse is charged per hour with a minimum of 30 minutes
24. **You need to resize a warehouse from X-SMALL to SMALL. How would you achieve this?**
    a. Execute ALTER WAREHOUSE SET WAREHOUSE_SIZE = 'SMALL'. New queries will run on the SMALL warehouse.
    b. Wait until all the queries have finished and execute an **`ALTER WAREHOUSE SET WAREHOUSE_SIZE = 'SMALL';`**
    c. Suspend the warehouse to prevent new queries and then **`ALTER WAREHOUSE SET WAREHOUSE_SIZE = 'SMALL';`**
25. **Query results are stored in the Result Cache for how long after they are last accessed, assuming no data changes have occurred?**
    a. 1 Hour
    b. 3 Hours
    c. 12 hours
    d. 24 hours

SnowPro: Model test 2

1. **True or False: A Virtual Warehouse can be resized while suspended.**
    a. True
    b. False
2. **Which of the following are valid types of Snowflake VIEW? (Choose any three).**
    a. REGULAR VIEW
    b. TABLE VIEW
    c. SECURE VIEW
    d. MATERIALIZED VIEW
    e. EXTERNAL VIEW
3. **Snowflake supports database replication only between Snowflake Accounts in the same Region.**
    a. True
    b. False
4. **Which of the following connectors are available in the Downloads section of the Snowflake Web Interface (UI)? (Choose two.)**
    a. SnowSQL
    b. ODBC
    c. R
    d. HIVE
5. **Which of the following terms best describes Snowflake's database architecture?**
    a. Columnar shared nothing
    b. Shared disk
    c. Multi-cluster, shared data
    d. Cloud-native shared memory
6. **Which of the following are valid approaches to loading data into a Snowflake table? (Choose all that apply.)**
    a. Bulk copy from an External Stage
    b. Continuous load using Snowpipe REST API
    c. The Snowflake Web Interface (UI) data loading wizard
    d. Bulk copy from an Internal Stage
7. **True or False: Some queries can be answered through the metadata cache and do not require an active Virtual Warehouse.**

a. True

b. False

8. **True or False: Each worksheet in the Snowflake Web Interface (UI) can be associated with different roles, databases, schemas, and Virtual Warehouses.**

a. True

b. False

9. **What data type should be used to store JSON data natively in Snowflake?**

a. String

b. Object

c. JSON

d. VARIANT

10. **Which Snowflake partner category is represented at the top of this diagram (Section 1)?**

Figure 15.1: *Snowflake – ecosystems tools and technologies*

a. Data Integration
b. Business Intelligence
c. Machine Learning and Data Science
d. Security and Governance

11. **A role is created and owns 2 tables. This role is then dropped. Who will now own the two tables?**
    a. The tables are now orphaned
    b. The user that deleted the role
    c. SYSADMIN
    d. The role that dropped the role
12. **Which privilege do you need to execute queries on a virtual warehouse?**
    a. OPERATE
    b. USAGE
    c. EXECUTE
    d. MODIFY
13. **Which of the following are valid Snowflake Virtual Warehouse Scaling Policies? (Choose two.)**
    a. Standard
    b. Economy
    c. Optimized
    d. Custom
14. **According to Snowflake best practice recommendations, which role should be used to create databases?**
    a. ACCOUNTADMIN
    b. SYSADMIN
    c. SECURITYADMIN
    d. USERADMIN
15. **Which of the following role is recommended to be used to create and manage users and roles? (Choose applicable)**
    a. ACCOUNTADMIN
    b. SYSADMIN
    c. SECURITYADMIN
    d. PUBLIC

16. **What are the different types of Snowflake Stages: (Choose three.)**
    a. Table Stage
    b. User Stage
    c. Named Stage
    d. Schema Stage
17. **The fail-safe retention period is how many days?**
    a. 1 day
    b. 7 days
    c. 45 days
    d. 90 days
18. **True or False: Reader Accounts incur no additional Compute costs to the Data Provider since they are simply reading the shared data without making changes.**
    a. True
    b. False
19. **True or False: Snowflake charges a premium for storing semi-structured data.**
    a. False
    b. True
20. **Which type of table corresponds to a single Snowflake session?**
    a. Transient
    b. Provisional
    c. Temporary
    d. Permanent
21. **Select the three types of tables that exist within Snowflake. (Choose three.)**
    a. Temporary
    b. Provisional
    c. Transient
    d. Permanent
22. **True or False: It is possible to load data into Snowflake without creating a named File Format object.**
    a. True
    b. False

23. **Which item in the Data Warehouse migration process does not apply in Snowflake?**
    a. Migrate Users
    b. Migrate Schemas
    c. Migrate Indexes
    d. Build the Data Pipeline
24. **What is the maximum compressed row size in Snowflake?**
    a. 50MB
    b. 16MB
    c. 8KB
    d. 1 TB
    e. 500 GB
25. **How frequently does Snowflake release new versions of software?**
    a. Every day
    b. Every month
    c. As needed.
    d. Every week

SnowPro: Model test 3

1. **Check all those are true about IDENTITY (Select 3)**
    a. Passing an identity check is called Authorization.
    b. Passing an identity check is called Authentication.
    c. Identity is about WHAT you can see/do.
    d. Identity is about WHO you are.
    e. Identity is sometimes tested and proven through username and password combinations.
    f. Identity is sometimes tested and proven through RBAC role assignments.
2. **Check all that are true about ACCESS(Select 3)**
    a. Proving a right to access something is called Authorization.
    b. Proving a right to access something is called Authentication.
    c. Access is about WHAT you can see/do.
    d. Access is about WHO you are.
    e. Access is sometimes tested and awarded through username and password combinations.

f. Access is sometimes tested and awarded through RBAC role assignments.

3. **Select all statements that are true about Snowflake's Access Control Model (Select 4)**
   a. Snowflake uses Role-based Access Control (RBAC)
   b. Snowflake uses Discretionary Access Control (DAC).
   c. In Snowflake, rights and privileges are awarded to USERS.
   d. In Snowflake, OWNERSHIP of items belongs to USERS.
   e. In Snowflake, rights and privileges are awarded to ROLES.
   f. In Snowflake, OWNERSHIP of items belongs to ROLES.
4. **Some roles have custodial oversight over objects owned by other roles. Check all that are true (Select 2)**
   a. If **`ACCOUNTADMIN`** creates a database, **`SYSADMIN`** can delete it.
   b. If **`SYSADMIN`** creates a database, **`PUBLIC`** can delete it.
   c. If **`SYSADMIN`** creates a database, **`ACCOUNTADMIN`** can delete it.
   d. If **`SECURITYADMIN`** owns a **`ROLE`**, **`ACCOUNTADMIN`** can change the name.
   e. If **`ACCOUNTADMIN`** creates a **`ROLE`**, **`SECURITYADMIN`** can delete it.
5. **Select all statements that are true about information schema?**
   a. The **`INFORMATION_SCHEMA`** schema holds a collection of views
   b. The **`INFORMATION_SCHEMA`** schema holds a collection of Tables
   c. The **`INFORMATION_SCHEMA`** schema cannot be deleted (dropped), renamed, or moved.
   d. The **`INFORMATION_SCHEMA`** schema can be deleted (dropped), renamed, or moved.
6. **There are four drop-menu settings or defaults for each worksheet. What are the four drop-menu worksheet context settings?**
   a. Database
   b. Schema
   c. Account
   d. Warehouse
   e. Role
   f. Password
   g. Username
7. **What is the role of the * symbol in a SELECT statement?**
   a. You can use it to quickly ask for all rows.
   b. You can use it to quickly ask for all tables.

c. You can use it to quickly ask for all columns.

d. You can use it to quickly ask for all schemas.

8. **What is the role of a LIMIT in a SELECT statement?**

   a. You can use it to quickly ask for a limited set of rows.

   b. You can use it to quickly ask for a limited set of tables.

   c. You can use it to quickly ask for a limited set of columns.

   d. You can use it to quickly ask for a limited set of schemas.

9. **Two of the context settings in a worksheet are in the upper-right corner, while two are in the middle, closer to the code. Why do you think they are positioned this way? Select 2.**

   a. The set near the code is intended for convenience and simplicity of code.

   b. The set near the upper corner is intended for convenience and simplicity of code.

   c. The set near the upper corner is NOT for convenience, it is required.

   d. The set near the code is NOT for convenience, it is required.

10. **You created a table and added some rows, later you try to query the table and Snowflake says the table does not exist? What is the first rule of Snowflake troubleshooting?**

    a. Check your warehouse.

    b. Check your user name.

    c. Check your role.

    d. Check your account.

11. **Scaling OUT/IN and scaling UP/DOWN are easily confused. Check all the true statements about scaling, below : (Select 3)**

    a. Editing a XS Warehouse and making it a M is an example of scaling UP.

    b. Editing a XS Warehouse and making it a M is an example of scaling DOWN.

    c. Editing a XS Warehouse and making it a M is an example of scaling OUT.

    d. Editing a XL Warehouse and making it a M is an example of scaling UP.

    e. Editing a XL Warehouse and making it a M is an example of scaling DOWN.

    f. Editing a XL Warehouse and making it a M is an example of scaling OUT.

    g. When an XS Warehouse automatically adds clusters to handle an increase workload, this is an example of scaling UP.

    h. When an XS Warehouse automatically adds clusters to handle an increase workload, this is an example of scaling DOWN.

i. When an XS Warehouse automatically adds clusters to handle an increase workload, this is an example of scaling OUT.

12. **Where does Snowflake store some of its Metadata?**
    a. The **`INFORMATION_DB`** database of each account.
    b. The **`METADATA_SCHEMA`** of each database.
    c. The **`INFO_METADATA`** schema of each database.
    d. The **`INFORMATION_SCHEMA`** schema of each database.
13. **Snowflake is cloud agnostic, especially when it comes to loading file from stage into Snowflake accounts. Check the true statements below(Select 3)**
    a. A Snowflake Account on AWS can have stage objects that point to, and load files from AWS, Azure and/or GCP.
    b. A Snowflake Account on AWS can only have stage objects that point to, and load files from AWS S3 buckets.
    c. A Snowflake Account on Azure can only have stage objects that point to, and load files from Azure Blob Storage.
    d. A Snowflake Account on Azure can have stage objects that point to, and load files from AWS, Azure and/or GCP.
    e. A Snowflake Account on GCP can have stage objects that point to, and load files from AWS, Azure and/or GCP.
14. **Which of these objects has the most compute power?**
    a. A Snowflake Database.
    b. A Snowflake Schema.
    c. A Snowflake Sequence.
    d. A Snowflake Warehouse.
    e. A Snowflake Data Mart.
15. **Which of these objects is the lowest or smallest in the storage container hierarchy?**
    a. A Snowflake Database.
    b. A Snowflake Schema.
    c. A Snowflake Region.
    d. A Snowflake Account.
    e. A Snowflake Table.
16. **True or False: A 4X-Large Warehouse may, at times, take longer to provision than a X-Small Warehouse.**
    a. True
    b. False

17. **What is the lowest Snowflake edition that offers Time Travel up to 90 days? Select one.**
    a. Standard Edition
    b. Premier Edition
    c. Enterprise Edition
    d. Business Critical Edition
18. **Select true statements about Stages from the options below (Select 4)**
    a. External Stages can be on any of the 3 major cloud providers (AWS, Azure, GCP)
    b. Internal Stages must be on the same cloud as your Snowflake Account.
    c. External Stages point to cloud folders inside the Snowflake Account.
    d. Internal Stages point to cloud folder outside the Snowflake Account.
    e. External Stages (or the folders they point to) can be loaded using the cloud provider's available tools or via the browser.
    f. Internal Stages (or the folders they point to) can be loaded using PUT command
19. **Which file types below are considered STRUCTURED? (Select 3)**
    a. Comma Separated (CSV)
    b. Avro
    c. Parquet
    d. Tab Separated (TSV)
    e. Pipe-delimited and named *.txt
20. **Which file formats support just data loading process in Snowflake? (Select Appropriate)**
    a. Comma Separated (CSV)
    b. Avro
    c. Parquet
    d. XML
    e. ORC
    f. JSON
21. **Which file formats support data loading and unloading process in Snowflake? (Select Appropriate)**
    a. Comma Separated (CSV)
    b. Avro
    c. Parquet

d. XML
e. ORC
f. JSON

22. **What is the Snowflake Object that helps us communicate with Snowflake about the structure of our data?**
    a. SCHEMA
    b. STAGE
    c. FILE FORMAT
    d. TASK
23. **Comma Separated files uses Commas as the symbol for column separation. What is another name for "Column Separator"?**
    a. Row Parser
    b. Field Delimiter
    c. Record Delimiter
    d. Row Separator
24. **Increasing the size of a Virtual Warehouse from an X-Small to an X-Large is an example of: Select one.**
    a. Scaling rhythmically
    b. Scaling max
    c. Scaling out
    d. Scaling up
25. **What are Materialized Views, External Tables and Iceberg Tables generally used for?**
    a. To load data into Snowflake quickly.
    b. To unload data from Snowflake quickly.
    c. To provide Snowflake access to data that has not been loaded.
    d. To extract, transform, and load (ETL) data into Snowflake.

SnowPro: Model test 4

1. **What is the definition of Metadata?**
    a. Data about meta
    b. Data that is above other data
    c. Data about data

2. **What is the default File Format used in the COPY command if one is not specified?**
    a. CSV
    b. JSON
    c. Parquet
    d. XML
3. **True or False: Fail-safe can be disabled within a Snowflake account.**
    a. True
    b. False
4. **What command is used to list files from the stage?**
    a. SHOW
    b. LIST
    c. COPY INTO
5. **What operations are allowed on databases like SNOWFLAKE_SAMPLE_DATA (inbound share data)?**
    a. CREATE / DROP / ALTER TABLE
    b. CREATE / DROP / ALTER SCHEMA
    c. SELECT with JOIN
    d. SELECT with GROUP BY
    e. INSERT INTO
    f. MERGE
6. **What do you think is the difference between a table and a view?**
    a. A table has more rows than a view.
    b. A table has fewer columns than a view.
    c. A view can capture a complex select statement and make it simple to run repeatedly.
    d. A view can capture a table's metadata, duplicate it, and transpose it in the future.
7. **What are the names of the 3 Snowflake Sharing Technologies?**
    a. Direct Sharing
    b. Materialized Views
    c. Snowflake Data Marketplace
    d. Private Exchanges
    e. Data Lakes

8. **Which of the following are NOT possible for SHARED data?**
    a. Run an `UPDATE` statement on the data
    b. Use a `REPLACE` function in a `SELECT` statement
    c. Add the data to your `OUTBOUND SHARE`
    d. Create a table in the `INBOUND SHARE`'s database
    e. Use a `GROUP BY` clause in a `SELECT` statement
    f. `JOIN` the `SHARED` data to your own data
9. **True or False: You cannot share a share.**
    a. True
    b. False
10. **If credits cost $3/hr, how much will it cost to run a Medium warehouse for 1 hour?**
    a. $3.00
    b. $6.00
    c. $9.00
    d. $12.00
    e. $14.00
    f. $16.00
11. **If credits cost $3/hr, how much will it cost to run a 3XL warehouse for 4 hours?**
    a. $12.00
    b. $24.00
    c. $36.00
    d. $192.00
    e. $256.00
    f. $768.00
12. **There are two options for Monitor Type when setting up the Resource Monitor. What are they?**
    a. Storage
    b. Warehouse
    c. Serverless
    d. Account
    e. User

13. **Which of the following statements are true? (Select 4)**
    a. Budgets are available in all Trial Accounts created after June of 2023.
    b. Budgets are only available in Paid Accounts.
    c. Once a Trial Account has been converted to paid, Budgets will be activated.
    d. Once a Trial Account has been converted to paid, Budgets will need to be activated.
    e. Budgets and Resource Monitors are the same thing, just named differently.
    f. Budgets are needed in addition to Resource Monitors because they include projections.
    g. Budgets are needed in addition to Resource Monitors because they cover more usage types.
14. **What is a Reader Account? (Select 3)**
    a. Another name for a Managed Account.
    b. Another name for a Managed User.
    c. An Account generated by a Snowflake Customer Organization for use by a non-Customer Org.
    d. An Account non-Customer Orgs to use to get access to data stored in Snowflake.
15. **What are some configurable options on Resource Monitors? (Select 4)**
    a. Monitoring the entire account or a single warehouse.
    b. Monitoring the entire account or a single database.
    c. The number of Storage Credits allowed.
    d. The number of Compute Credits allowed.
    e. Monitoring Reset Period (Daily, Weekly, Monthly, Yearly).
    f. Suspensions and Notifications based on percentages of use.
16. **Shares take place between which two groups?**
    a. Data Providers & Consumers
    b. Full Accounts & Reading Accounts
    c. Sharers & Consumers
    d. Data Writers & Data Readers
17. **Data Consumers can be set up to consume data using which two types of consuming accounts?**
    a. Sharing Accounts & Query Accounts
    b. Query Accounts & Full Accounts

c. Sharing Accounts & Providing Accounts

d. Full Accounts & Reader Accounts

18. **The Snowflake Data Marketplace has two types of listings. What are they? (Select 2)**

    a. Business Critical and Premier

    b. De-Personalized and Personalized

    c. Standard and Business Critical

    d. Standard

    e. Personalized

    f. De-Personalized and Premier

19. **Select all the true statements below.**

    a. Listings are either Standard or Personalized.

    b. Data Sets are either free and immediately available, or available by request via a personalized process flow.

    c. Free data sets are always customized.

    d. Personalized data sets can be customized.

    e. Free data sets are never customized and personalized data sets are sometimes customized, but not always.

20. **What are some criteria required of a data set for it to qualify for listing on Marketplace? (Select 4)**

    a. Complies with Privacy and Other Laws

    b. Fresh and Not Static

    c. Recently Created Sample Data

    d. Legally to Distribute

    e. English Language

    f. Real (not mocked up data)

21. **Select the available system-defined roles in Snowflake similar ACCOUNTADMIN? (Select 4)**

    a. PRIVATE

    b. TESTROLE

    c. ORGADMIN

    d. SYSADMIN

    e. SECURITYADMIN

f. USERROLE

g. USERADMIN

22. **Find all the true statements about Snowflake Warehouses below and select them. (Select 3)**

    a. Snowflake Warehouses are like Data Marts in that they hold subsections of an organization's data.

    b. Snowflake Warehouses were designed to hold data in structures based on Kimball and Inmon Data Warehousing theories.

    c. Snowflake Warehouses do not hold data.

    d. Snowflake Warehouses provide computing power to run queries, load data, and carry out other tasks.

    e. Functionally, a Snowflake Warehouse is more like a spread sheet workbook than a laptop CPU.

    f. Functionally, a Snowflake Warehouse is more like a laptop CPU than a spreadsheet workbook.

23. **Which statements are true about Snowflake-managed Stages? (Select all that apply)**

    a. They do not require you to have your own separate cloud account.

    b. You can use web browser tools to load files into them.

    c. They are also called "Internal Stages"

    d. They must be/are on the same cloud as your Snowflake account.

24. **If we do not tell Snowflake anything about our file structure, what will it presume about the structure? (Select 3)**

    a. The file data is flat.

    b. The file data is nested/hierarchical.

    c. The data rows are separated by New-Record-# symbols.

    d. The data rows are separated by Carriage Return/Line Feed symbols.

    e. The data columns are separated using commas.

25. **Which Snowflake object enables loading data from files as soon as they are available in a cloud storage location? Select one.**

    a. Pipe

    b. External Stage

    c. File Format

    d. VARIANT

Answer keys

The following are the answers to the model test questions:

Model test 1

Please refer the answer keys for the Model test 1:

1	a, c, d
2	a
3	b
4	d
5	b
6	d
7	a
8	a, b, c
9	a, b
10	a
11	a
12	a, b, c, f
13	a, b, c, d, e
14	a
15	a, b, c, d
16	a, b, d
17	a
18	a, b
19	d
20	a, b, c
21	b
22	a
23	b
24	a
25	d

Model test 2

Please refer the answer keys for the Model test 2:

1	a
2	a, c, d
3	b
4	a, b
5	c
6	a, b, c
7	a
8	a
9	d
10	a
11	d
12	b
13	a, b
14	b
15	c
16	a, b, c
17	b
18	b
19	a
20	c
21	a, c, d
22	a
23	c
24	b
25	d

Model test 3

Please refer the answer keys for the Model test 3:

1	b, d, e
2	a, c, f
3	a, b, e, f

4	c, d
5	a, c
6	a, b, d, e
7	c
8	a
9	a, c
10	c
11	a, e, i
12	d
13	a, d, e
14	d
15	e
16	a
17	c
18	a, b, e, f
19	a, d, e
20	b, d, e
21	a, c, f
22	c
23	b
24	d
25	c

Model test 4

Please refer the answer keys for the Model test 4:

1	c
2	a
3	b
4	b
5	c, d
6	c
7	a, c, d
8	a, c, d

9	a
10	d
11	f
12	b, d
13	b, d, f, g
14	c
15	a, d, e, f
16	a
17	d
18	d, e
19	a, d, e
20	a, b, d, f
21	c, d, e, g
22	c, d, f
23	a, c, d
24	a, d, e
25	a

Conclusion

In this chapter, we had multiple set of sample questions along with the answer keys to help the readers practice what they can expect in the SnowPro Exam. Hope this helps the individuals to analyze their performance and better prepared for the exam using these sample questions.

Good luck to your exam preparation!!!

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Index

D

E

Snowflake SnowPro Core Certification Guide COF-C02

DESCRIPTION

Snowflake, a revolutionary cloud data warehouse platform, has gained immense popularity due to its scalability, performance, and ease of use. This comprehensive guide is designed with the knowledge and skills necessary to pass the SnowPro Core Certification exam and excel in the world of Snowflake.

Prepare for the SnowPro Core Certification with this all-inclusive guide. Understand Snowflake's architecture, data types, and security features while mastering virtual warehouse management. Learn essential data movement strategies and performance optimization techniques like caching and clustering, and explore the latest Snowflake features. This guide includes mock tests and expert advice to help you confidently tackle the certification exam. You will gain a solid understanding of Snowflake's unique strengths, including its ability to manage both structured and semi-structured data, secure information, and optimize performance for complex tasks.

By the end of this book, you will be well-prepared to tackle the SnowPro Core Certification exam with confidence. You will have a solid grasp of Snowflake's fundamentals, be able to write efficient SQL queries, optimize performance, and implement best practices for data security and governance.

KEY FEATURES

- Covers all essential SnowPro Core Certification topics.
- Includes real-world scenarios and use cases.
- Reflects the latest Snowflake features and best practices.
- Includes practice tests to prepare for the certification.

WHAT YOU WILL LEARN

- SnowPro Core Certification overview, including subject area/domain breakdown.
- Essential tips to prepare and pass the exam.
- Snowflake's fundamentals to advanced key concepts outlined in the SnowPro Certification.
- Industry best practices and recommendations on various key concepts.
- SnowPro Core Certification sample practice questions to test the overall knowledge.

WHO THIS BOOK IS FOR

This book is for current and aspiring emerging data professionals, data/solution architects, data engineers, database administrators, data analysts, data scientists, and anyone who wants to explore and learn about the modern data cloud platform.

BPB PUBLICATIONS
www.bpbonline.com

ISBN 978-93-5551-888-0