Snowflake Essentials

Getting Started with Big Data in the Cloud

Frank Bell
Raj Chirumamilla
Bhaskar B. Joshi
Bjorn Lindstrom
Ruchi Soni
Sameer Videkar

Apress®

Snowflake Essentials

Getting Started with Big Data in the Cloud

Frank Bell
Raj Chirumamilla
Bhaskar B. Joshi
Bjorn Lindstrom
Ruchi Soni
Sameer Videkar

Apress®

Snowflake Essentials: Getting Started with Big Data in the Cloud

Frank Bell
Encino, USA

Raj Chirumamilla
Monmouth Jct, USA

Bhaskar B. Joshi
Bethesda, MD, USA

Bjorn Lindstrom
Eatonville, USA

Ruchi Soni
New Delhi, Delhi, India

Sameer Videkar
Magdeburg, Sachsen-Anhalt, Germany

ISBN-13 (pbk): 978-1-4842-7315-9
ISBN-13 (electronic): 978-1-4842-7316-6
https://doi.org/10.1007/978-1-4842-7316-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484273159. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors

Frank Bell is a SnowPro, Snowflake Data Hero, entrepreneur, and data thought leader who has been working with databases since 1994 when he first started with Oracle in the United States Air Force. He ran the consulting firm IT Strategists from 1999 to 2019 and built one of the fastest-growing Snowflake practices in 2018, which was sold to Fairway Technologies and then acquired by Accenture. He now runs IT Strategists (ITS) Cloud Products (https://itstrategists.com) and Snowflake Solutions (https://snowflakesolutions.net), which are focused on building Snowflake tools, such as Snoptimizer (https://snoptimizer.com) (Snowflake Performance, Cost, and Security Optimization). He also leads the Accenture Snowflake West Market Unit. He has lived, breathed, and eaten Snowflake since early 2018 and believes Snowflake and the data cloud are one of the top game-changing technologies of this decade.

Raj Chirumamilla is a Snowflake Certified Architect and AWS Solutions Architect Associate with 20+ years of IT experience. He is an experienced, hands-on solutions architect working in the data and analytics area, helping customers migrate to cloud and build cost-effective and highly available solutions and data pipelines with modern data architecture frameworks. He has been working with Snowflake's cloud data platform (CDP) for more than three years.

Bhaskar B. Joshi is a Snowflake Certified Architect with 20 years of experience in managing data on various platforms. He has been working with Snowflake for over three years in multiple industries. He is currently a hands-on data architect helping customers create scalable solutions using modern data architecture frameworks in the cloud.

Bjorn Lindstrom is an IT veteran with 40 years of experience in many aspects of computer technologies. He has held roles in software development, software quality assurance, database administration, and large-scale big data systems and is an experienced solutions architect. He was one of the first to be certified in Snowflake.

Ruchi Soni is a technology leader and multi-cloud enterprise architect. Her work lies in helping customers accelerate their digital transformation journey to the cloud and build next-generation apps on forward-looking platforms such as Snowflake. She brings extensive experience in planning, architecting, building, and scaling future-ready platforms that are highly available and agile. Ruchi is SnowPro Core certified, Google Cloud certified, and AWS certified and is an Accenture Certified Master Data Architect and Senior Technology Architect.

Sameer Videkar is a data expert with more than 12 years of experience working on data platforms, enterprise data warehouse systems, master data management, data migration, and ETL implementations. He is a certified data architect with additional certifications in Snowflake, Agile Scrum, and PMP.

Foreword

In 2010, Marc Andreessen wrote an op-ed in *The Wall Street Journal* titled "Why Software Is Eating the World" as many brick-and-mortar businesses were being disrupted by digitally native firms like Amazon, Uber, and Apple. The pace of disruption has accelerated further as organizations have embraced a data-driven approach to transforming businesses. Data has become central to transforming how companies interface with their customers, suppliers, and partners, find new ways to generate revenues, create new ecosystems, and improve operational efficiencies.

Data is now used in innovating almost every aspect of life, such as new drug research, autonomous vehicles, smart cities, adjusting insurance claims, creating art and music… and the list goes on. Data has become so pervasive that the best way to describe this new world is that "Data is *feeding* the world." The recent COVID pandemic has also necessitated that companies move their apps and data estates to the cloud. The combined data and cloud trend has given rise to a new breed of data platforms that enable enterprise initiatives to become data-driven. One such popular data platform is Snowflake – a platform that has taken the industry by storm since its unprecedented IPO in September 2020.

This is a timely book on the Snowflake data platform that covers the essentials as well as some expert topics such as data architectures. Frank, Bjorn, Raj Chirumamilla, Sameer Videkar, Bhaskar Joshi, and Ruchi Soni are top experts in Snowflake and have packed this book with stellar content on Snowflake and how to use it effectively.

The authors have become deep Snowflake experts and have been working extensively on Snowflake for many years. They have helped build Snowflake consulting solutions that have helped numerous Snowflake implementations and clients. Frank, in particular, has been immersed in Snowflake since the beginning of 2018. He transformed his previous big data consulting business to become one of the key Snowflake implementation partners before being acquired by Accenture.

This book will help data professionals learn all the essentials around the Snowflake Data Cloud. It also covers some of the transformative Snowflake architecture and features in depth such as data sharing. My favorite chapter in the book is Chapter 14 on data sharing, exchanges, and marketplaces. It provides an insightful look at how data sharing is transforming the evolution of data collaboration to digitally transform businesses.

—Shail Jain

CHAPTER 1

The Snowflake Data Cloud

Snowflake has been one of the most transformative data technologies I've come across in my 30-year technology career. Over the last several years, Snowflake has disrupted the big data and analytical relational database management system (RDBMS) industries. The creation of a Software as a Service (SaaS) cloud database built entirely on multiple public clouds has been an analytical database game changer. The Snowflake Data Cloud with an almost unlimited scale has been a revolutionary improvement in both data processing ease and scale.

Big Data Cloud History

As the Internet continued to grow in the decade of the 2000s, data creation and collection grew at a rapid pace. Amazon introduced S3 in March 2006 as part of Amazon Web Services (AWS). S3 was a great way to store files in the cloud, but it didn't have any traditional data management capabilities. The next month in that same year, the Apache Software Foundation introduced a new big data technology named Hadoop. Hadoop for quite some time became the go-to big data solution. Many of us who were data professionals at the time worked to implement Hadoop solutions, but initially Hadoop was very complex and required developers with coding knowledge to get any value out of it. Hadoop did not have any SQL interface until 2010. Hive was only introduced on October 1, 2010, and then it still wasn't really well integrated with the entire Hadoop solution.

Snowflake Beginnings

Realizing there were still incredibly difficult challenges scaling big data solutions in 2012, the Snowflake founders came together to build a relational database management system (RDBMS) built from the ground up on a cloud architecture.

F. Bell et al., *Snowflake Essentials*, https://doi.org/10.1007/978-1-4842-7316-6_1

NoSQL solutions including Hadoop were cloud technologies that had come a long way since 2006, but Hadoop especially was still expensive and too complex for most organizations to operate. The Snowflake founders, Benoit, Thierry, and Marcin, were the first technologists to completely rethink and rearchitect an RDBMS to work with cloud-based technologies. This new data architecture created a major differentiation in speed and scale, ease of use, and ease of data sharing that led to Snowflake's rapid customer adoption and business growth. These fundamental technical differentiators created foundational business differentiators for customers. These business differentiations were significant enough to overcome any business and technology switching costs related to moving to the Snowflake Data Cloud.

In this chapter, we will cover what the Snowflake Data Cloud really is and why you can benefit as both a business professional and data professional from learning the Snowflake essentials. We introduce to you the overall Snowflake Data Cloud and the transformative impact it is having on the overall world of data sharing.

Why the Snowflake Data Cloud Is Different

Snowflake was the first database architected to run on the cloud from the ground up. Snowflake prefers to brand itself as the Snowflake Data Cloud since June 2020, but it was created to be a SQL database engine with automated scaling and tuning at first. This is what made Snowflake so disruptive and separated it from traditional analytical RDBMSs, on-premise massively parallel processing (MPP) solutions, and its early cloud competitors.

Google BigQuery, AWS RedShift, Microsoft Azure, and even Databricks were all created from totally different architectural foundations and with different initial purposes. RedShift was a modification of the PostgreSQL database that initially was not architected to separate compute and storage. RedShift was still the first cloud analytical database, so it still had a first mover advantage, but its foundations are built off an existing on-premise RDBMS technology. Google BigQuery was built on top of Dremel technology. BigQuery was initially designed as a black box query engine, not an RDBMS. Databricks was created as the enterprise version of Apache Spark, an open source distributed computing framework.

Snowflake differentiates itself from all of these other solutions since its core architecture was an RDBMS built for the cloud and initially created by two Oracle RDBMS veterans who took all of their advanced RDBMS engineering knowledge to

create a fully scalable cloud database system. This is crucially important because the core of how a system works and grows comes from its foundational purpose and architecture (assuming it stays true to its foundational architecture).

Snowflake's Unique Architecture

Due to some of the original underlying architecture, Snowflake was able to expand beyond the concept of just a single database management system. Readers coming from RDBMSs understand the sheer pain we used to have to deal with connecting on-premise database systems to even different databases that ran on the same database systems such as Oracle and SQL Server.

Snowflake's architecture is a hybrid model of both a shared-disk and a shared-nothing architecture. At the core of Snowflake's architecture are three separate layers that we will go into more depth in Chapter 3. Here is a quick overview of them:

- Cloud Services: The cloud services layer of Snowflake handles all of the services within the database such as metadata management, authentication, security, and query optimization.
- Compute: Snowflake has virtual warehouses that run the compute. The query layer is separated from the disk storage.
- Storage: Snowflake uses micropartitions, which are heavily compressed and optimized to organize the data into a columnar data store. The data is stored within the cloud provider's cloud storage (e.g., S3 in AWS). Compute nodes connect to the storage layer to retrieve the data and process it.

Figure 1-1 shows a visualization of the Snowflake Data Cloud's three layers: cloud services, compute, and storage.

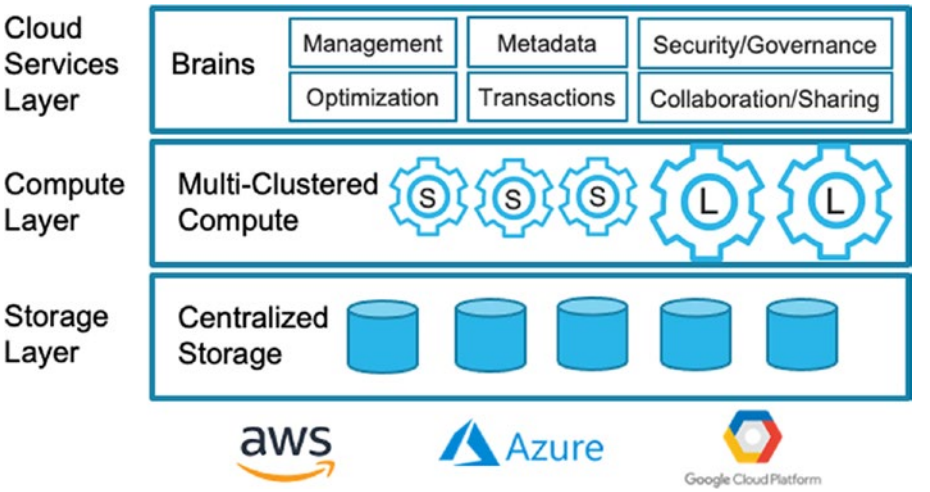

Figure 1-1. *Snowflake's Three-Layer Architecture*

Snowflake's Unique Platform Features

There are five fundamental Snowflake Data Cloud features enabling Snowflake to be so disruptive and fundamentally different from previous database solutions:

- The separation of compute from storage
- Automated data maintenance and scaling
- Ease of use
- Speed. Speed. And more speed
- Data sharing

Let's go through these fundamental differences one by one.

The Separation of Compute from Storage

The main architectural decision to separate compute from storage provided Snowflake with major differentiation from all of its competitors at the beginning besides Google BigQuery. This enabled Snowflake to come to market with a true pay-as-you-go RDBMS data service. At the time, this was simply amazing and unheard of. It was the first

cloud RDBMS where you only had to pay for what you consumed. This allowed small and mid-size companies to access large compute for reasonable costs. It also enabled unprecedented scaling of compute to solve data challenges.

Any customer of Snowflake in the world could bring *massive* compute to any data problem for a few seconds or minutes at a reasonable cost vs. spending months negotiating and installing the standard RDBMS big data solutions, such as Netezza-, Teradata-, or Hadoop-based solutions. This was fundamentally revolutionary for a data engineer or data entrepreneur who was comfortable with SQL. We could be querying and analyzing large datasets within the same day with our Snowflake account. This was a game changer.

Cloud-Backed Availability

Also, this enabled a distributed architecture of availability across availability zones and removed the immediate need of all previous on-prem solutions for backup. This architecture also enabled time travel and cloning features, which were revolutionary concepts to bring to an RDBMS. (Since 2018, BigQuery and Databricks have copied the basics of Snowflake's features.)

Cloud-Enabled Scale

From an engineering perspective, what should not go unnoticed is Snowflake's micropartition architecture, which is really a continuous write structure in cloud data storage. This fundamentally creates non-locking data retrieval from storage files. This type of architecture allows for massive scale and finally allows data to only have one single source.

This is a key point of differentiation. The Snowflake Data Cloud overall brings scale unlike on-prem solutions and even other data solutions based on the cloud. Snowflake's architecture created a solution that can share data without making copies. Copies of data (including Datamarts) were a necessary architecture in the past so that data performance could scale to meet business needs. The problem is that copies of data necessitate more maintenance and create more complexity, organization, and governance challenges and costs. One of the largest business data problems across enterprises still is inconsistent data and inconsistent analysis due to analyzing different copies of data.

Snowflake Compression

Another fundamental engineering work from Snowflake was the proprietary and sizeable compression. Faster compression and less data transfer led to less Input Output (IO) costs and an overall faster architecture. It is not uncommon to see 5x compression when migrating data to Snowflake.

Automated Data Maintenance and Scaling

One of the major differentiators of Snowflake that still remains today is the fundamental change of making database maintenance, management, and scaling a business function vs. an engineering function. From 1994 to 2018, I spent too much time learning every single trick on how to optimize data architectures. It was fun to be a technical hero to come into a situation where the RDBMS was not scaling and place a few indexes and speed everything up dramatically, but it really is something that could be engineered into a database system to be automated.

Fundamentally, I assume we engineers had blinders on and overlooked that we had the performance metadata for years to create self-indexing and self-optimizing database systems. Also, for all the data professionals reading this, star and snowflake schemas and Ralph Kimball's growth in popularity came about only due to technical scale limitations. The creation of Hadoop was driven by similar limitations of big data scaling as well. RDBMSs that people loved and were comfortable using needed database administrators (DBAs) too often to maintain speed and scale even for mid-size workloads as more and more data was created, stored, and analyzed. Also, as data became bigger and bigger, traditional RDBMSs and even on-premise MPP revolutionary solutions at the time like Netezza, Teradata, and Exadata couldn't scale either. Snowflake was the first data solution to embrace internal indexing and scaling for the analytical database. This was another game changer and fundamentally changed the maintenance cost structure for organizations by removing the bulk of complexity and the people maintenance costs of DBAs and data engineers to scale basic growth and reporting RDBMSs.

Ease of Use

One of the key features to technologies that are adopted rapidly is the ease of use of the technology. Data professionals find Snowflake very easy to use if they come from any of the traditional RDBMS or MPP database backgrounds. The founders really designed Snowflake to be an easy-to-use analytical database by removing cumbersome administrative burdens, establishing all features around consistent DDL and DML SQL standard syntax, and making it very easy to join and share data.

SQL Is Well Known

By using the RDBMS common SQL language as the core data retrieval mechanism, Snowflake made it much easier for the millions of data professionals proficient with SQL to interact with and understand their offering. SQL has been around since 1974 and is used by millions of people across the world. Many more people know SQL than Python and other programming languages. This has enabled SQL data professionals to adopt the platform more easily vs. other solutions initially not based on ANSI SQL.

Joining Enterprise Data Is Easy

With on-premise databases, it was often a challenge both administratively and with performance to join data from datasets within different databases. Again, this often resulted in making copies of data and transferring it even for internal purposes. Overall, this just created more friction and work for data professionals to get things done. Snowflake removes that barrier by allowing data querying and analysis on one primary set of data. That data table never has to be copied. This is another game changer.

Viewing Past Versions Is Built In

Time travel and Zero Copy Cloning make it easier to look at your database as it existed at any point in the past of your choosing. We will go into time travel and Zero Copy Cloning in more detail later, but just realize these fundamentally changed how data professionals worked. If you made a mistake before with traditional RDBMSs, then you often would have to restore a backup and fix your mistake. With a few lines of time travel code, you can easily go back to the version of your table before you made your mistake. This enables data professionals to fix data errors within minutes vs. hours or days.

Zero Copy Cloning created the ability to truly move toward Agile Data Warehousing. For the first time with big data, data professionals were able to clone production-size data to perform full-scale production data–level tests.

Speed. Speed. More Speed Please

In the beginning of 2018, I took Snowflake to one of my long-standing clients. They had been running this massive Athena job that was taking like a month to complete. We moved a subset of that equivalent job that was taking over 24 hours over to Snowflake during a proof of concept. It finished in less than 1 hour, and we ported it over in a couple days. A more than 24x improvement without any tuning and optimizing! Thousands of Snowflake customers repeated this type of nirvana over the past several months and years. New Snowflake customers often see massive speed improvements of 2–10x++ that create massive differentiation from its data processing competition.

Data Sharing

Over the last 20+ years, data professionals used many different mechanisms to perform data sharing within organizations and across organizations. We mainly used copying techniques to copy data from one database or data warehouse to another. Constraints surrounding these techniques created entire businesses and engineering solutions around how to use architectures, tools, etc. to copy data and scale data usage and data movement.

Many data professionals still unfortunately have not been exposed to this incredibly more efficient solution of no-copy data sharing provided with Snowflake. One of the best overall features of the Snowflake Data Cloud and a Snowflake account by itself is how easily you can connect different datasets (tables, views) to any Snowflake database or schema within your account that you have access to. Data sharing is also being improved so you can easily replicate data from one account within one region and cloud provider to another region and another cloud provider.

Timeline of the Snowflake Data Cloud Creation

The story around the architecting of Snowflake goes back to 2012. The following are some of the key dates in Snowflake's development that you may want to be aware of and that might help you in talking with and selling your own clients on the platform:

- Founded - July 23, 2012
- Raised Series A Round of $5M led by Sutter Hill Ventures – August 2012
- Bob Muglia appointed as CEO – June 2014
- Raised $26M additional funds – October 2014
- Came out of stealth mode with 80 customers on AWS – October 2014
- Raised an additional $45M and launched its first product, the cloud data warehouse – April 2017
- Raised Series $100M of additional funds –
- Launched on Azure,
- raised $263M of additional funds at a 1.5B valuation - July 12, 2018
- Frank Slootman joined as CEO – May 2019
- Launched on Google Cloud Platform (GCP) – June 4, 2019
- *Launched the Snowflake Data Exchange (which eventually became the Snowflake Data Marketplace) – June 2019
- Raised another $479M in a round led by Dragoneer Investment Group – February 7, 2020
- Snowflake IPO (Initial Public Offering), which raised $3.4B, making it the largest software IPO – September 16, 2020

Summary

On September 16, 2020, Snowflake became the largest software Initial Public Offering (IPO) in history. It continues to gain additional customer adoption, and Snowflake engineering and analytics resources are scarce. If you learn these essentials provided within this book, you will be in a good position to solve Snowflake development tasks and be hired for Snowflake work.

The trend is clear that companies and entire industries are moving to the cloud. As a data professional who wants to keep up, it is essential that you continue to develop your skills with cloud databases. Snowflake currently is one of the most popular cloud databases, and you can benefit by learning how to use the essentials to make yourself more valuable for database professional jobs.

CHAPTER 3

Snowflake Data Cloud Architecture

This chapter will cover the essentials of the Snowflake Data Cloud architecture that has made Snowflake widely popular. This hybrid architecture provides Snowflake with ease of use as well as fast and scalable performance. When the founders decided to build a new relational database completely based on the cloud, they were able to create architectural advantages beyond existing database architectures. One of the key architectural beliefs they were founded on was that tying storage to compute created challenges with scaling on the cloud.

In this chapter, we will cover how Snowflake's decision to have a hybrid architecture of traditional shared-disk and shared-nothing architectures has helped Snowflake create a powerful and highly scalable RDBMS solution. Snowflake capitalizes on using a central data repository similar to a shared-disk architecture for persisted data within each cloud provider. At the same time, it processes queries using MPP (massively parallel processing) similar to shared-nothing architectures. Snowflake uses compute clusters to do this where each node in the cluster stores a part of the dataset locally. This hybrid approach provides both data management simplicity and improved performance of the scale-out architecture.

The Snowflake Data Platform as a Cloud Service

Snowflake has introduced us to DSaaS (Data Software as a Service), which runs on their data platform as a cloud service. This simply means there is no software or hardware to install, configure, or manage. There are no software upgrades to manage either. Snowflake Corporation manages all of that complexity for you.

F. Bell et al., *Snowflake Essentials*, https://doi.org/10.1007/978-1-4842-7316-6_3

Snowflake takes care of the ongoing maintenance of your database and the tuning, querying, security, and management services related to it. This also means that you, the data professional, now have access to a full RDBMS delivered in the cloud that is optimized for scale. This really changed the landscape for organizations to not have to invest in continuous maintenance of hardware and software. It frees the data professional from having to deal with a lot of scaling engineering that was required in the past with almost all other data systems. Snowflake's unique architecture also provides a true cost and speed advantage for organizations by removing lots of underlying database administrator maintenance costs and database planning costs. Since Snowflake runs completely on the cloud, every component of Snowflake's DSaaS runs on cloud infrastructure within each cloud provider (besides the optional connectors, drivers, and command clients). In this chapter, we will cover the three main architectural layers of Snowflake's Data Software as a Service.

Caution While the removal of most DBA activities is truly amazing and game changing (Hooray! No more index management and maintenance!), all enterprise-level database organizations and professionals who are processing terabytes to petabytes of data will quickly realize that with great data power still comes great responsibility. The ease of use that the Snowflake Data Cloud provides will still require either automated or some administrative human management of access, data cloning, data usage, data quality, and data governance. I mean it is awesome that you can load terabytes of data into Snowflake and process it quickly. In order to maintain high-quality data, an organization must use professional third-party services or customized Snowflake functions or tools to do resource monitoring and data governance. Snowflake data warehouses, lakes, and clouds can become out-of-control data swamps and cost-control nightmares if you do not set up and continually monitor your Snowflake account. [You have received fair warning here!!!]

Big Data Architecture History

Before Snowflake, the main two big data architectural approaches were shared nothing and shared disk. Figure 3-1 shows a visualization of the two different architectures. Shared nothing is when the data is partitioned and processed across separate server nodes. Each node has the sole responsibility for the data it has. The data is completely segregated. Shared disk is basically the opposite where all data is available to all the nodes. Any of the nodes can write to or read from any part of the data it wants.

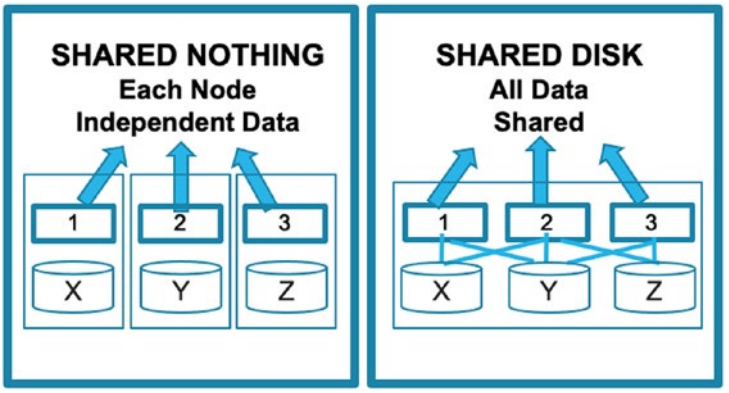

Figure 3-1. *Shared Nothing vs. Shared Disk*

Snowflake's Hybrid Architecture

Snowflake's architecture is a hybrid of both the shared-nothing and shared-disk architectures. It is set up to take advantage of benefits of both concepts. The key is that Snowflake separates storage and compute, which gives it great flexibility in both scaling processing up and out and allowing for consumption-based pricing.

The Data Warehouse Evolution

Data warehouse technology and analytical databases have been evolving over the last 30 years from RDBMSs to MPP on-premise systems. Then Hadoop and cloud data analytical systems evolved over the last ten years. This eventually evolved to Snowflake coming out with its groundbreaking architecture separating compute from storage initially on the AWS cloud provider. Figure 3-2 illustrates this evolution.

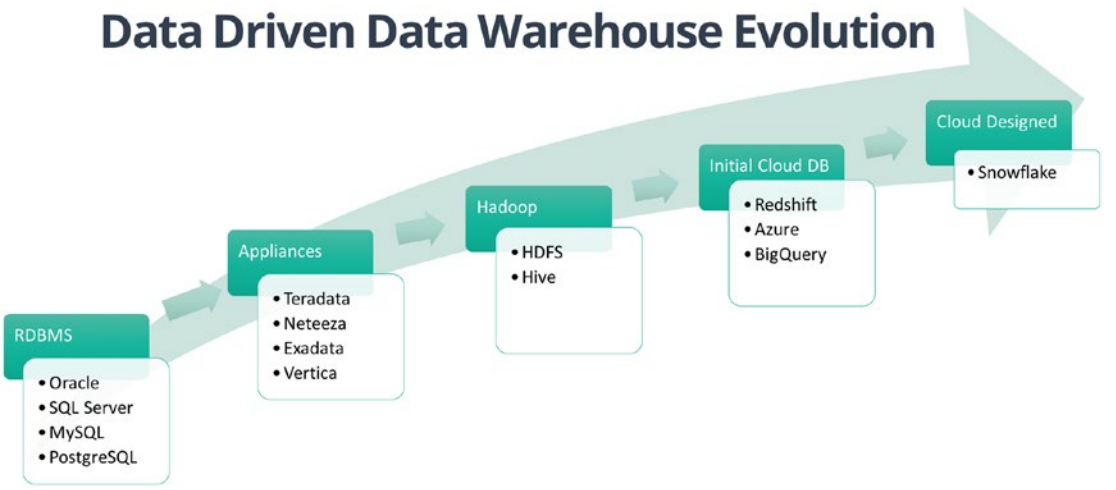

Figure 3-2. *Data Warehousing Architectural Evolution*

We saw this evolution happening with our consulting solutions over the last 20+ years, and I wrote an article explaining the overall data warehouse evolution in more depth here: www.linkedin.com/pulse/data-warehousing-evolution-frank-bell.

The Snowflake founding team saw both the migration to the cloud and the challenges related to all the existing solutions of RDBMSs, MPP systems, Hadoop, NoSQL, and initial cloud databases where the initial architecture was not from a cloud provider architectural foundation. The Snowflake founders published this white paper, which covers the fundamentals of Snowflake's architecture beliefs in the early days:

https://dl.acm.org/doi/10.1145/2882903.2903741

Figure 3-3 below shows an overview of Snowflake's three layered architecture.

Snowflake's Layered Architecture

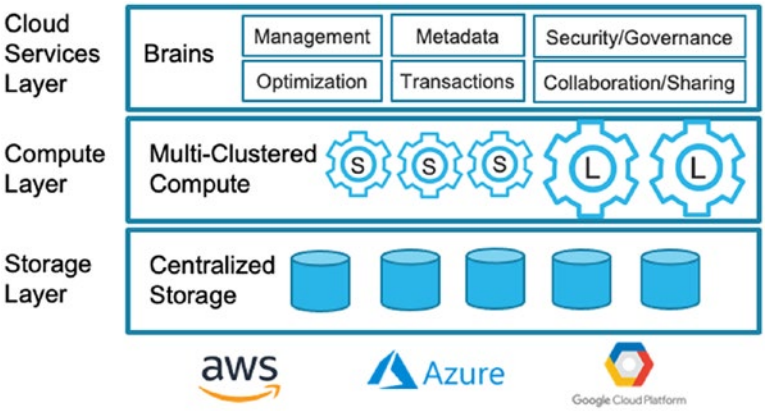

Figure 3-3. *Snowflake's Three Layers of Cloud Services, Compute, and Centralized Storage*

Let's dive into how each of these three independent layers works.

Cloud Services Layer

The cloud services layer is really the brains behind the Snowflake Data Cloud. It provides the main services of the Snowflake Data Cloud that all users interface with including

- Optimization services
- Management services
- Transaction services
- Security and governance services
- Metadata services
- Data sharing and collaboration services

This layer controls all the authentication and security to create centralized security and better data governance. One of Snowflake's key features is that it transparently exposes the query history, and this is done through this layer as well. The services layer also handles all of the metadata management, query optimization, and Snowflake's data sharing services (which we will discuss in depth in Chapter 14).

Compute Layer

Snowflake's compute layer is fully separated from the other two layers of cloud services and storage. This compute layer runs "virtual warehouses," which can be of various T-shirt sizes and run different query workloads independently and concurrently. At a high level, these virtual warehouses are MPP compute clusters with multiple nodes of CPU and memory.

This allows organizations to have different workloads on different Snowflake warehouses. For example, one warehouse can be focused on loading data, while another warehouse is handling queries for data analysts. A Snowflake customer can technically run tens or hundreds of separate independent warehouses running different workloads and never contending for the same compute resources. These workloads can even be accessing the same exact data at the same time with no contention or bottlenecks of traditional databases. All the provisioning of this virtual compute node is done by Snowflake depending on the selections of the end user around virtual warehouse size and cluster size between one and ten nodes. All these virtual warehouses (and there can be hundreds or thousands of them if your business needs it) work completely independently.

Storage Layer

Snowflake's storage layer is completely separated from the compute layer, which also allows Snowflake to charge a more reasonable cost for storage than previously seen in the database offerings when both compute and storage were tied to each other. Snowflake's storage layer cleverly leverages the native blob storage capabilities of a cloud provider. (On AWS it uses S3. On Azure it uses Azure Blob. On Google it uses Google Cloud Storage.) This is done through a columnar database architecture, which has raw files compressed and encrypted within the native cloud storage available. This layer is designed to provide sub-second query response times with centralized data at petabyte scale. Snowflake also leverages their raw micro-partition storage technology which we will discuss in more depth later. Heavy compression on the storage layer is also used to improve performance. We often see three to five times compression with data loaded into Snowflake.

The Separation of Storage from Compute

The major architectural advantage Snowflake harnessed from the cloud was the separation of compute from storage. This is the bottleneck that every on-premise system would run into as data continued to grow bigger and bigger. Even with optimized hardware, on-premise systems at some point just could not keep up with the massive scale of cloud-based provider server farms.

This separated architecture also enabled Snowflake to deliver to customers a "pay for what you use" offering. This traditionally has been a business architecture winning formula because now for the first time even small startups could afford this pay-as-you-go architecture as they worked within investment funding to achieve product market fit and scale their data and business.

Micropartitions and Their Use in Snowflake

Micropartitions are another one of Snowflake's key architectural concepts designed to work well in a cloud architecture. The main benefits from using them are the speeds at which most workloads can be delivered compared with on-premise or other cloud RDBM systems that used traditional indexes or hardware optimizations. Snowflake automatically divides and groups rows of tables into these compressed micropartitions of 50–500 MB of data. Figure 3-4 shows an example of three micropartitions. Micropartitions are immutable physical files that are automatically partitioned based on ingestion order unless you set up auto-clustering to define how partitions should be set up. Ideally the micropartitions are clustered (sorted) as efficiently as possible to allow for **pruning**. (See Figure 3-5.) These micropartitions allow the Snowflake engine to easily replicate segments evenly for distribution across nodes. These micropartitions also are part of the architectural design that allows Snowflake to handle datasets of any size (even petabytes) since they cleverly distribute them into these micropartitions with metadata automatically and continuously updated on them.

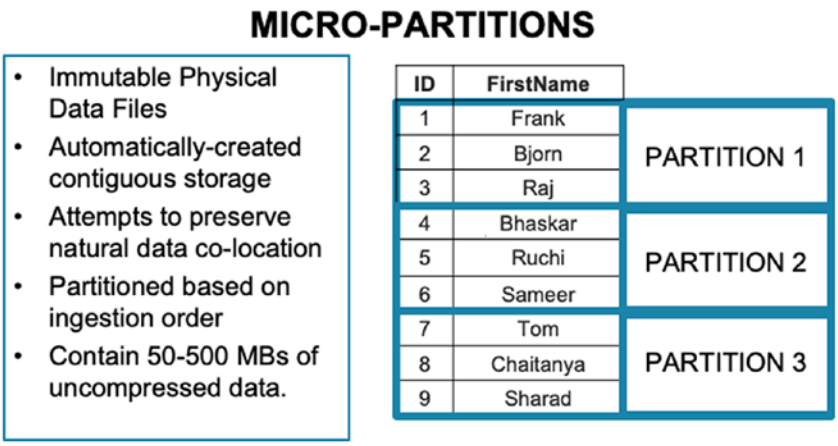

Figure 3-4. *Snowflake's Micropartitioning Example*

How Pruning Works

Query pruning is mainly what it sounds like. A database architecture is constructed to use a query optimizer that prunes away micropartitions unnecessary to run a query. This optimizes the Input Output (IO) overhead, compute, and overall work required and makes queries much faster if they only need to access a small subset of partitions vs. scanning them all. Snowflake's metadata is continuously updated and enables Snowflake's query optimizer to precisely prune columns at query runtime. This is really neat since it enables just-in-time pruning and optimization based on the micropartition metadata, which is continuously updated. Snowflake also architected this to work on semi-structured data like JSON and XML.

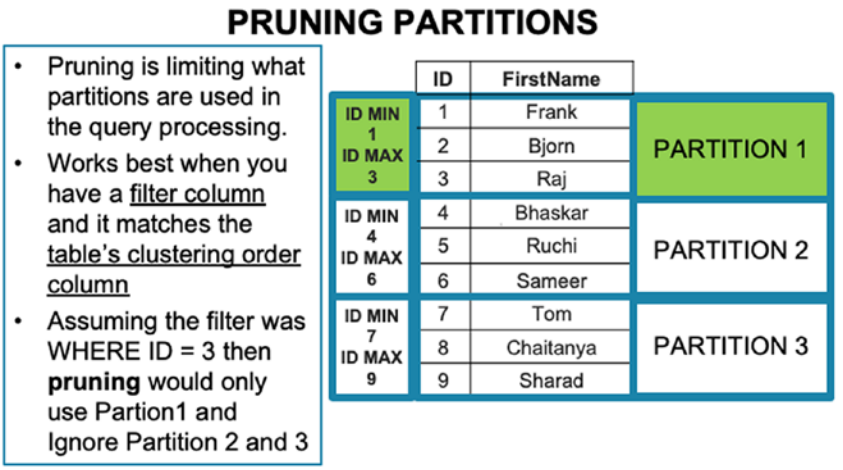

Figure 3-5. *Snowflake's Partition Pruning Example*

Cluster Keys

Cluster Keys and Automated Clustering in Snowflake

All Snowflake editions automatically cluster your data with default cluster keys when the data is ingested into tables. Typically, this is done on columns of temporal data types such as date and timestamp since this is a natural load sequence for any time series–type datasets. The reality is though that not all workloads are time-based ordered. Some tables are sequenced on some type of primary ID or joint set of columns, which organizes the sequence of the dataset within a table. Snowflake suggests that when you have tables larger than 1 TB, you need to define optimized cluster keys and enable auto-clustering. This will help if your table continues to have ingestion organization different than your workloads or Data Manipulation Language operations (UPDATE, DELETE, MERGE, etc.) that reorganize your data in non-optimal micropartitions. Just realize that while automated reclustering has many benefits including ease of maintenance and non-blocking organization, it comes at a credit consumption cost.

Tip When you define multi-column clustering keys for a table with the CLUSTER BY clause, then the best practice is to order columns from lowest to highest cardinality.

When you load data, if you order the data before loading on the keys or filters that you will be using, then you can make the overall database system run more efficiently. This will also save you compute costs if you have auto-clustering enabled on the cluster keys because the rows are already preordered so there isn't much additional auto-clustering required. If your data is initially loaded and distributed in the order it will be queried, it is common sense that this will provide you better optimization. You are basically pre-organizing and ordering the dataset for your workloads.

Reclustering for Optimization

Reclustering is just reorganizing micropartitions based on your cluster keys. In a way it's like re-indexing or reorganizing files so that the metadata and the partitions themselves are highly optimized for pruning based on your historical workloads. Clustering and reclustering in Snowflake is now fully automatic. (If you see references to manual clustering, that has been disabled.)

Snowflake's Caching Architecture

One of my favorite features when I was initially introduced to Snowflake was that they would cache query results for 24 hours and not charge customers for accessing those query results. When you or another user **in your account** initiates the same exact query the second time, it returns instantaneously for no additional cost. I really thought this was a customer-first type of offering to not charge the end customer additional costs if Snowflake themselves did not incur costs. If you have hundreds of users doing the same exact query, you are saving tons of extra duplicative workloads that have both a hard cost and energy/climate cost.

As we discussed previously, Snowflake operates three independent redundant layers. The centralized storage layer is the cold base storage. This layer is optimized with the micropartition architecture and pruning discussed previously. Let's cover how the layers of caching and storage work together to achieve optimal performance.

The following are in Snowflake storage and caching layers:

> Result Cache: Snowflake caches the results of every query executed within the last 24 rolling hours. This cache is available to any other user on the same account who executes the same exact query if the underlying data has not changed.

Local Disk Cache: The virtual warehouse compute layer optimizes a separate cache as well when it is activated to retrieve and compute data operations. For example, on Snowflake AWS, each of the EC2 instances has RAM and an SSD disk. When a user runs a query, the data is retrieved from the centralized cold storage (S3) into the EC2 instance in both memory and SSD. Since Snowflake uses columnar and smart micropartitions, it will typically retrieve a limited number of columns that will be cached in the SSD disk. It is a smart limited cache based on workload patterns. This creates a warm cache that executes many regular predicted workloads extremely fast since retrieval from memory and SSD is much faster than the centralized blob storage.

Remote disk (S3, Azure Blob, Google Cloud Storage): This is where the raw compressed Snowflake micropartition files are stored.

The Benefits of Cloning

One of my most favorite features of Snowflake is the capability to clone databases within seconds. By introducing this feature, Snowflake finally allowed data engineering professionals to do truly Agile Data Warehousing. Before this feature, Agile Data Warehousing really was a misnomer when dealing with big data. Even the largest Fortune 100 companies typically did not want to pay to create a duplicate test and staging environment with on-premise or other cloud databases that required copying data. Also, when copying tens or hundreds of terabytes of data, it always is as fast as the amount and IO (Input Output) you have available, so it is less agile.

Cloning really enabled the data warehouse and big data professionals to replicate what had been working in agile software development for years. We discuss Snowflake cloning in more depth in Chapter 9.

Tip The actual time to clone a database is dependent on the amount of metadata objects it has. Most small to mid-size databases with a few hundred objects can be cloned within a minute. Databases with large numbers of objects in the thousands will take minutes to clone.

Performance Optimization Features

As of this writing, there are three main standard optimization features in production within Snowflake. We will briefly touch on them here, and they will be covered in more depth in later chapters. While Snowflake is much more powerful than previous RDBMSs, it still can be optimized for performance. Currently, Snowflake does optimize queries with a query optimizer. Snowflake's standard performance on most data structures smaller than 1 TB even without clustering or ordering is still amazingly fast. When tables get larger than 1 TB though, the distribution of cluster keys will improve performance. Also, there is an additional query optimization service, which is in private preview currently named the Query Acceleration Service. This service will allow for larger-scale scan-heavy workloads to be accelerated on specific warehouses where this is enabled. It will be best for speeding up performance with short needs for larger-scale queries so it doesn't impact the other workloads. It also will be able to accelerate long-running queries. Let's dig into the current performance optimization services available in production.

Materialized Views

Materialized views are somewhat standard in RDBMS architectures. They allow you to store frequently used aggregations and results to avoid recomputing and increase the speed of results. Snowflake's materialized views are actually limited compared with some other vendor offerings because they *only* support materializing a subsection of an existing table. They do not provide functionality for joining tables and materializing the results. Snowflake's materialized views also have additional costs.

Automated Clustering on Cluster Keys

Snowflake automatically reorganizes tables to work with query patterns. This allows your database to use pruning more effectively to process only relevant partitions from large tables. The end benefit is faster queries and lower compute costs for the queries. There is a cost for the background clustering maintenance to re-cluster new data as it comes into the table.

Search Optimization Service

Snowflake provides this service to enhance performance around pinpoint lookups of filtered values. This is a serverless function managed by Snowflake, which creates equality predicates. This service supports data types of numbers, date, time, strings (exact match), and binary (exact match).

Summary

The Snowflake Data Cloud consists of three separate layers of cloud services, compute, and storage. Snowflake's usage of caching layers enables some of Snowflake's powerfully fast performance. Snowflake's architecture also enables groundbreaking features such as time travel and cloning, which are enabled by the micropartitions and the separation of compute from storage. The Snowflake architecture continues to add additional cloud services such as materialized views, automated clustering, and search optimization. We hope you enjoyed this chapter explaining the essentials of the Snowflake Data Cloud architecture.

CHAPTER 4

Snowflake Web Interface: Classic Console

In this chapter, we will cover Snowflake's Classic Console web interface and all the functionality within it. We will guide you to where all the key features are on the web interface. Snowflake's Classic Console is a well-thought-out web interface that has been a key part of the Snowflake platform since the beginning. As of this writing, the Classic Console is still the main interface used to interact with Snowflake. However, Snowflake Corporation has recently re-released their new front-end interface named the Preview App (Snowsight), which will be the future interface eventually. We also cover the Preview App (Snowsight) in the next chapter.

Web Interface: Classic Console Main Overview

The Snowflake Classic Console is the main way you can interact with the Snowflake Data Cloud. It provides access to almost all the functionality of the entire Snowflake Data Cloud service at both an account and an organization level. You can get access to the main areas of Databases, Shares, Data Marketplace, Warehouses, Worksheets, History, and Account (with the ACCOUNTADMIN role or equivalent). Before we cover the Classic Console, one major tip that you must always be aware of is that the interface changes are dependent upon what role you have selected in the upper right. In Figure 4-1 is the Snowflake Classic Console interface with the upper-right role selection displayed.

F. Bell et al., *Snowflake Essentials*, https://doi.org/10.1007/978-1-4842-7316-6_4

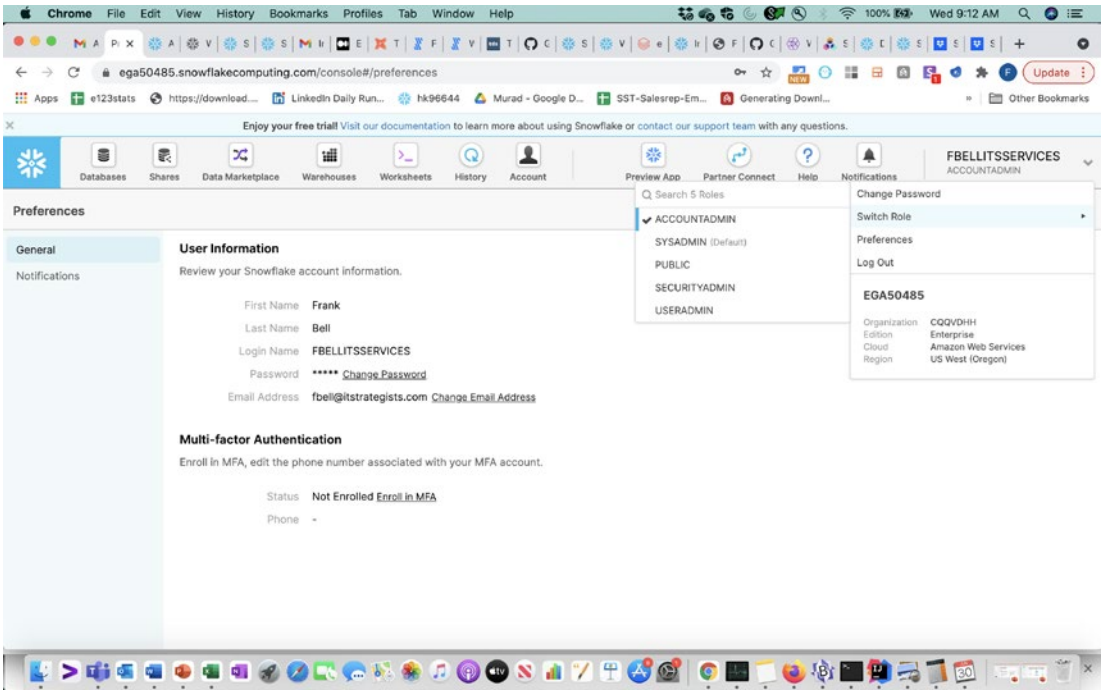

Figure 4-1. *Classic Console and Role Selection*

Tip The "role" you select in the upper right of the preceding web interface impacts what icons are displayed on your web interface. Unless you are using the ACCOUNTADMIN role (or an equivalent custom role with Account privileges), then you will not see the Account icon, and you will not be able to access any of the Account details in the Classic Console. You also will not see the Notifications icon or be able to create shares or view them without additional access.

Databases

The Databases icon is the main area for all database objects in the Snowflake Data Cloud. The main database interface contains the functionality to Create, Clone, Drop, and Transfer Ownership on databases within the Snowflake Data Cloud.

Figure 4-2 shows what the Snowflake Classic Console database listing interface looks like. You can easily create, clone, drop, or transfer ownership on databases from the buttons in the following interface. If you are coming from other relational databases, you will notice how easy it is to create a database without dealing with any complex configuration settings.

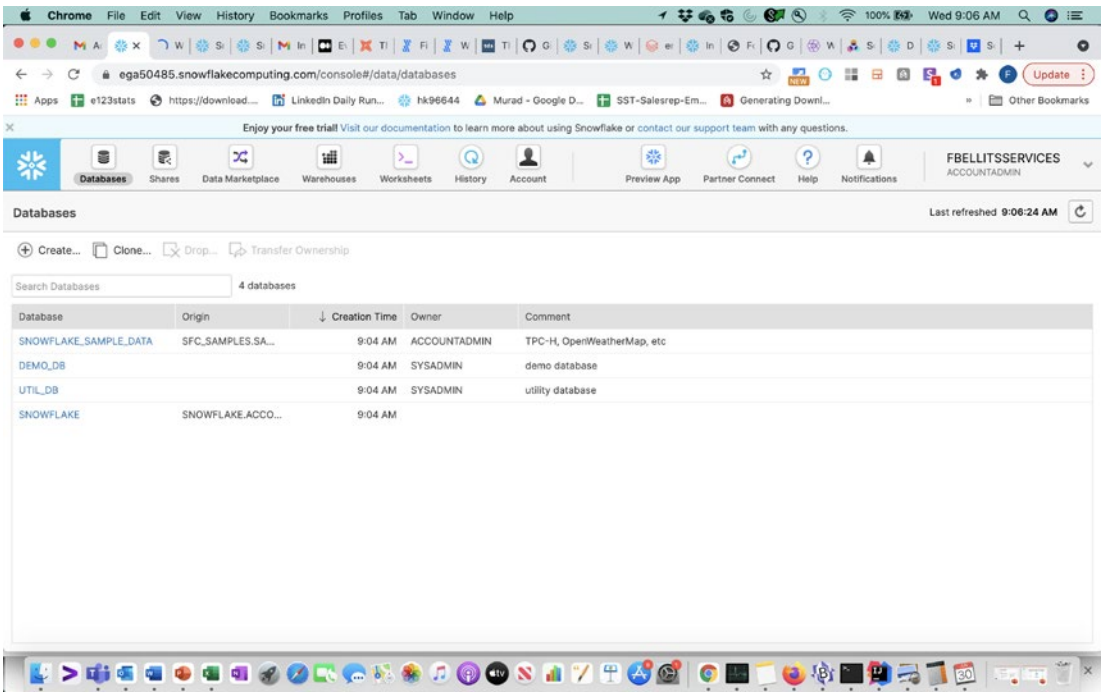

Figure 4-2. *Databases*

You may notice at first in the interface that the Create and Clone links are active and the Drop and Transfer Ownership links are grayed out. The Snowflake Classic Console is consistent in graying out actions that are not available unless you highlight the row of the listed object.

For you to view all the navigation to objects within a database such as tables, views, schemas, etc. in the Classic Console, you need to click a database name on the left-hand side of the database listings. Once you click the database name, you will see the navigation change to display tabs for the main seven objects within databases: Tables, Views, Schemas, Stages, File Formats, Sequences, and Pipes. You will also notice the navigation displays which database you have selected with the Databases ➤ [Database Name] in the interface. Let's go through each of these database objects and how to navigate to them and when you want to use them in your Snowflake Data Cloud work.

Tables

Tables are the key construct of all relational database systems. They are the mechanism that contains the data. Snowflake tables are easy to use if you come from any previous relational database work since they are created by normal Data Definition Language (DDL) syntax of CREATE, ALTER, DROP, TRUNCATE, and RENAME statements. The only major difference with Snowflake tables related to DDL is their VARIANT data type. Otherwise, all of their Data Definition Language and data types are standard to relational database systems. In Figure 4-3 is the Snowflake Classic Console table listing interface. This figure shows two rows of two tables created in the CITIBIKE demo database, which will be the database we use for examples later in this book.

You can Create, Create Like, Clone, Load Table, Drop, and Transfer Ownership on tables from this web interface. We find creating tables typically is easier to do in DDL within worksheets, but if you are a GUI person, you may like to occasionally create them in the web interface, but this doesn't really scale to large amounts of tables.

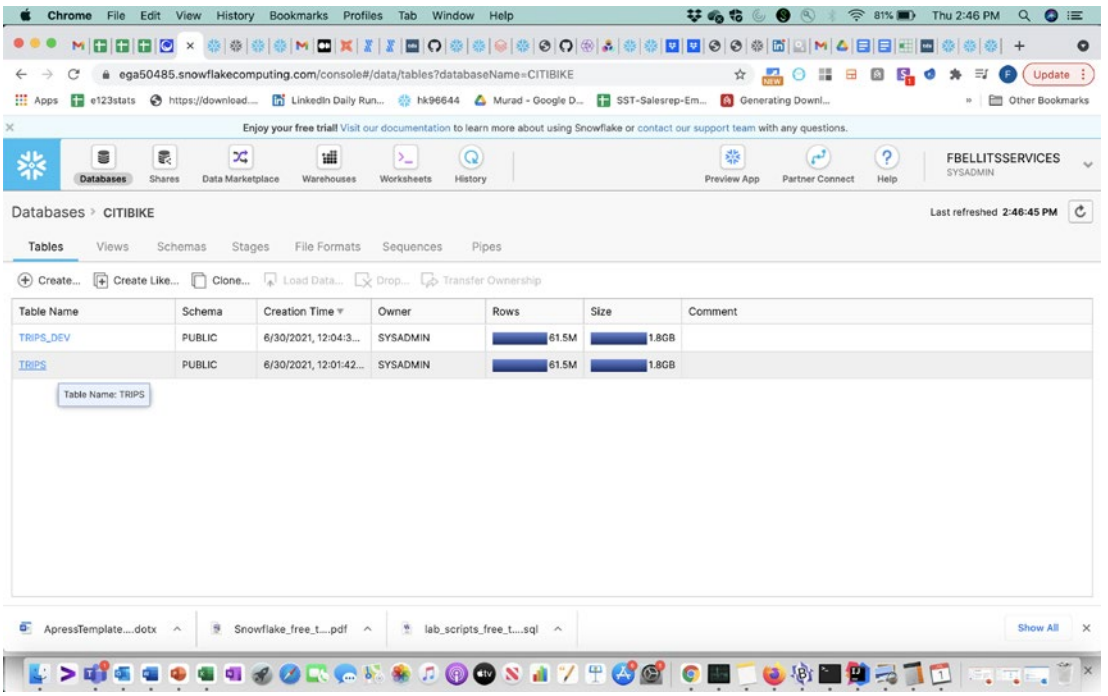

Figure 4-3. *Tables*

You will also notice again that some of these functions are grayed out. Once you click within the row of the table, then they will become active like how the database object listings worked.

Views

Views are another key construct of relational database systems, allowing users to create virtual and materialized views of physical table data. The main purpose of views in relational database systems is to provide additional security of data and allow for flexibility of view changes without having to move/copy data. Snowflake also provides a feature of secure views, which allow users to create views with the definition and the details of the view hidden from the end viewer/user for greater security. Secure views also prevent exposing underlying data to user-defined functions or other programmatic mechanisms. Secure views are the ONLY type of view you can use to share in Snowflake's data sharing feature. Secure views should not be used for views that do not need this level of privacy or security because Snowflake's query optimizer bypasses some optimizations used for regular views and there could be some level of query performance impacts with secure views.

Snowflake also provides their version of materialized views for Enterprise and above editions, which can help provide fast performance of queries by materializing the data within the view. Snowflake's version of materialized views can only be created from one table. You cannot join other tables to create a Snowflake materialized view.

A list of views is shown in Figure 4-4 where you can take actions such as Create, Drop, and Transfer Ownership of views. Like the table listing, you will notice that certain functions are grayed out that need to be done on one specific view. You must select a view in the list to have the Drop and Transfer Ownership links become active.

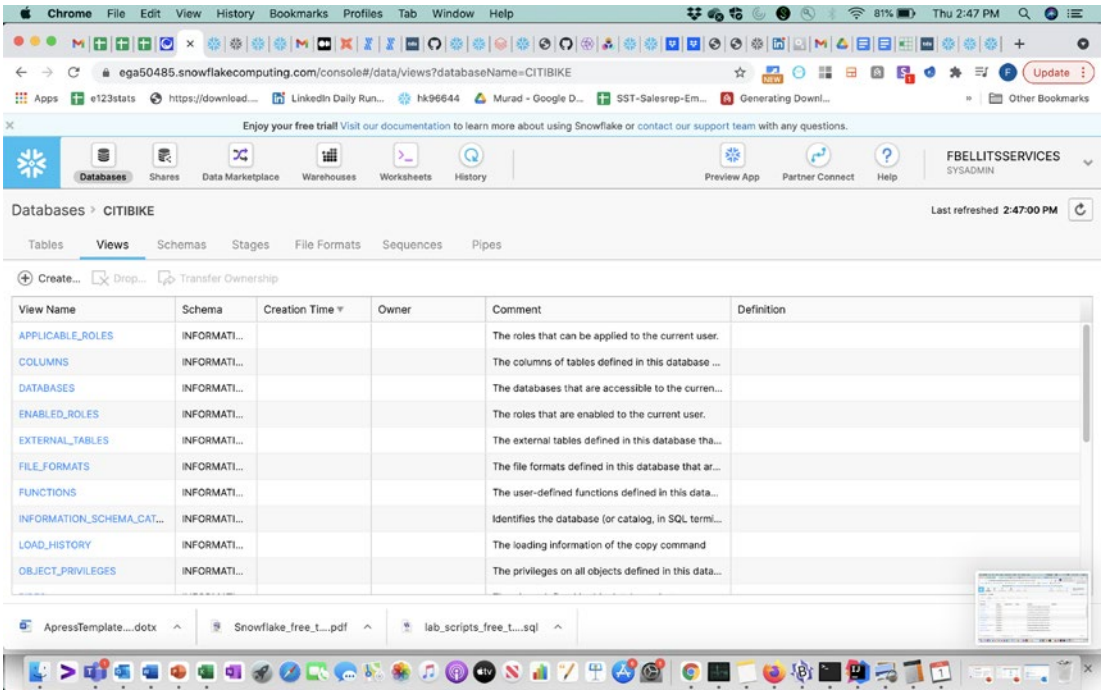

Figure 4-4. *Views*

Schemas

Schemas are a relational database mechanism created for organization and security. They are a common mechanism used within relational databases and work similarly to any relational database you have used in the past. Snowflake schemas are also enhanced with the key new Snowflake feature of cloning. Figure 4-5 shows a standard view of Snowflake schemas listed within a database.

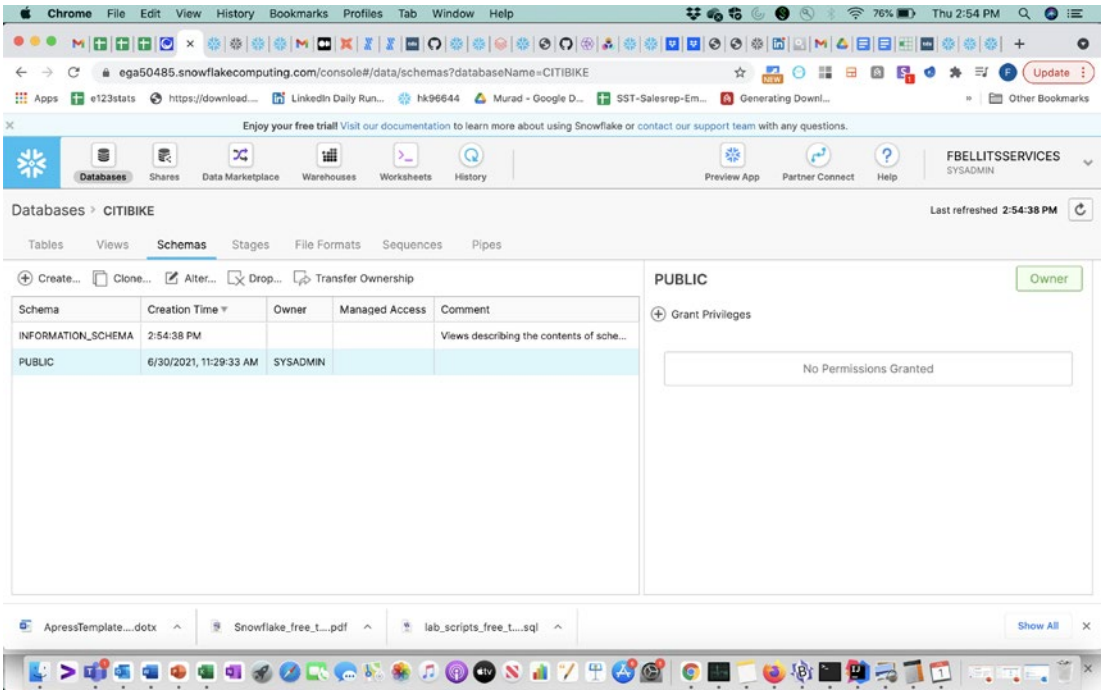

Figure 4-5. *Schemas*

Stages

Stages are a Snowflake Data Cloud concept and specific to Snowflake. All cloud databases require you to "stage" or really move the data from on-prem or other locations to an accessible cloud location. Snowflake's unique architecture allows you to have stages in four different ways, which include an "Internal" Snowflake Stage (technically on the AWS cloud but completely Snowflake controlled) and "External" Stages, which currently include Amazon S3, Microsoft Azure, and Google Cloud Platform.

Tip One key point to be very clear about is that when you transfer files to an Internal Snowflake Stage, you are charged for that file storage as a part of Snowflake's storage costs. We have seen many Snowflake users stage large amounts of files in Snowflake Stages and load them into the database and forget to purge them from the Internal Stage. If these files are sizeable, the storage costs

can add up and provide no real business value since the files have already been loaded into Snowflake. Our standard recommendation is to use the PURGE = TRUE option with COPY INTO code loading from an Internal Stage.

In Figure 4-6, the Create Stage functionality is shown where you choose between the four different types of stages including Snowflake Managed, AWS S3, Microsoft Azure, and Google Cloud Platform. You can stage your data on any of these even if your Snowflake account is running off another cloud provider. These are just staging locations where data can be staged to be loaded.

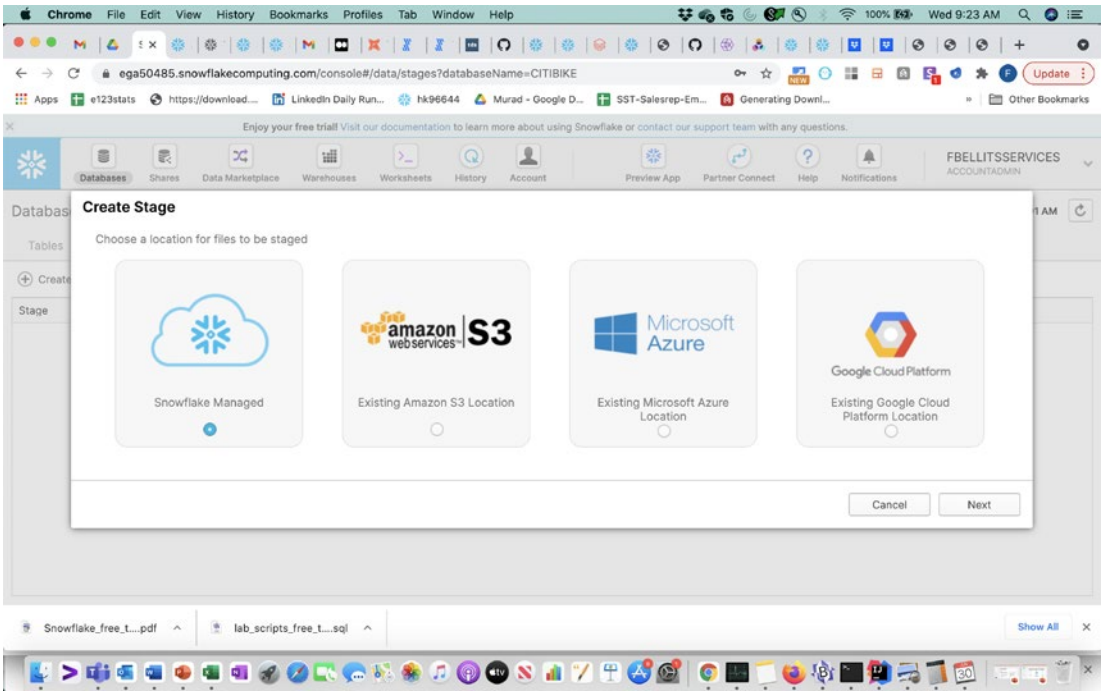

Figure 4-6. *Stages*

Let's understand how you create an External Stage on Amazon's AWS S3 file storage. Every External Stage needs a stage name and stage URL at a minimum. Most customers want to have encryption security as well for almost all use cases, so you can easily add encryption keys.

Figure 4-7 shows the interface you can use to add all of these elements of the AWS External Stage depending on your business and technical needs including Name, Schema Name, S3 URL, AWS Key ID, AWS Secret Key, Encryption Master Key, and Comment. I always recommend that you comment every object you create.

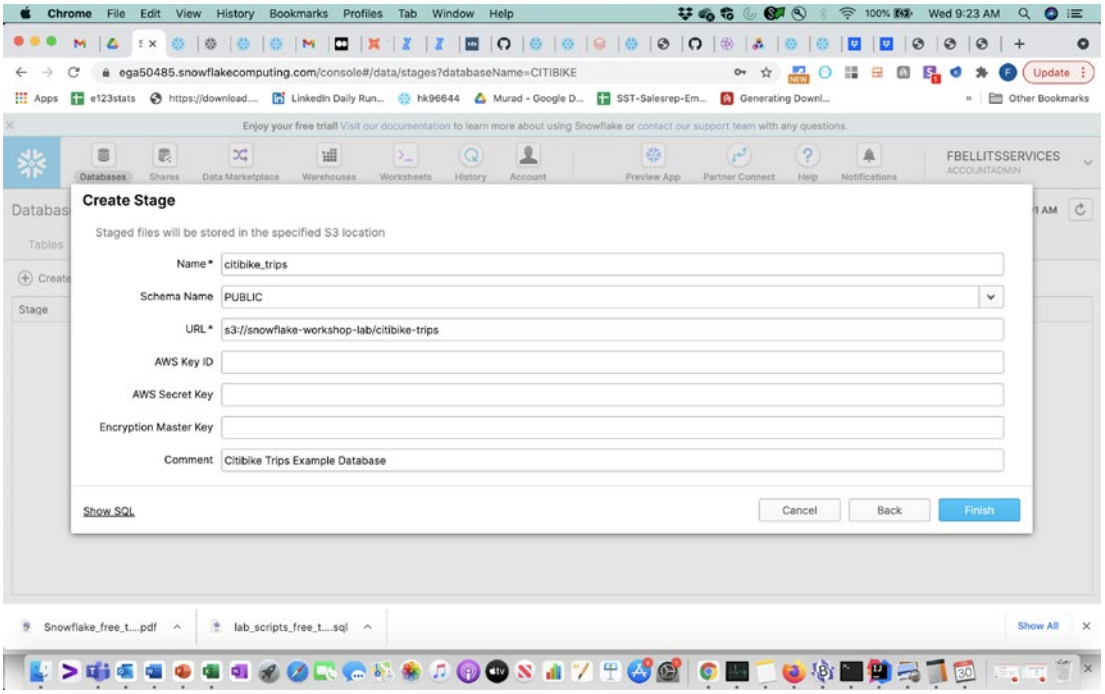

Figure 4-7. *Creating AWS Stage*

File Formats

File Formats similar to stages are specific to Snowflake. They are very similar though to relational database syntax that describes file type and format details so they can be loaded by bulk upload commands such as bcp (bulk copy program) for SQL Server, Oracle Loader for Oracle, nzload for Netezza, and FastLoad for Teradata.

File Formats simply describe the format of the file you are loading from a stage or directly from a COPY INTO statement. They can be dynamically defined within code as well. They provide the COPY INTO statement with the details of the file so it can be loaded correctly. You can Create, Clone, Edit, Drop, and Transfer Ownership of File Formats similar to other database objects.

File Formats have many different syntax options as shown in Figure 4-8. The figure is an example of the top of an empty form for creating a File Format that you will get when you hit Create File Format on the web interface.

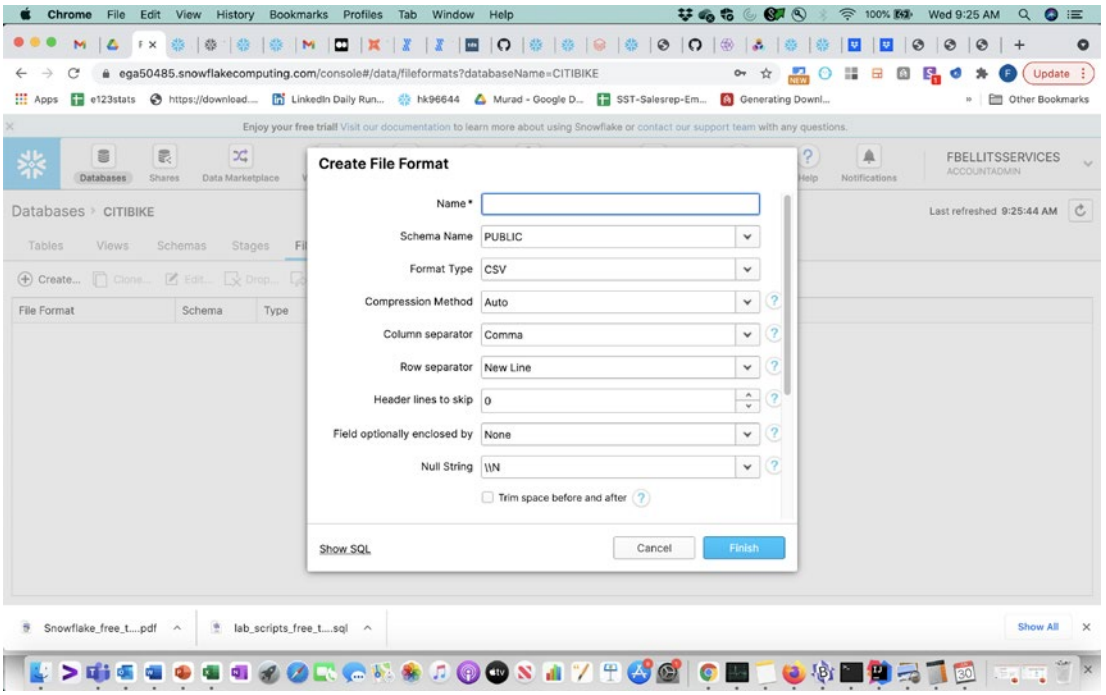

Figure 4-8. *File Formats*

The Create File Format form on the Snowflake interface provides several fields for you to fill out. The following are the field names and their descriptions:

- Name: Fill in the name of the File Format.
- Schema Name: Fill in the schema of the File Format that you are creating.
- Format Type: Available options are CSV (which DOES NOT specifically mean CSV, but really means Delimited File Type), JSON, XML, Avro, ORC, and Parquet.
- Compression Method: Available options are Auto, Gzip, Deflate, Raw Deflate, Bz2, Brotli, Zstd, and None.

- Column separator: Available options are Comma, Vertical Bar, Tab, None, and Other. You can add a custom column separator with the Other option, but it can ONLY be one character.
- Row separator: New Line, Carriage Return, None, or Other. You can add a custom row separator with the Other option.
- Header lines to skip: Enter the number of rows or lines (if any) that you want to skip.
- Field optionally enclosed by: None, Double Quote, or Single Quote. You use this to deal with the common delimited file issue of the delimiter being within the field or column. This encapsulates extra delimiters to enable the delimited data to load properly.
- Null String: Can be \\N, NULL, NUL, or Other. You can add a custom null string with the Other option.
- Trim space before and after checkbox: This enables the COPY INTO to trim white space before and after the field.
- Error on Column Count Mismatch: This is a key setting and typically for quality purposes is enabled. It identifies an error if the number of columns in the source does not match the number of columns in the destination.
- Escape Character: Backslash, None, or Other. You can add a custom Escape Character with the Other option. This is used to escape separators or special characters like single and double quotes.
- Escape Unenclosed Field: Backslash, None, or Other. You can add a custom Escape Unenclosed Field with the Other option.
- Date Format: Auto or Other. You can add a custom Date Format with the Other option.
- Timestamp Format: Auto or Other. You can add a custom Timestamp Format with the Other option.
- Comment: Enter a comment that describes the purpose or details of the File Format object.

Sequences

Sequences are another common database object in relational databases. Sequences allow users to increment and generate unique integer values for rows of data within tables. Figure 4-9 shows the view of the Create Sequence form.

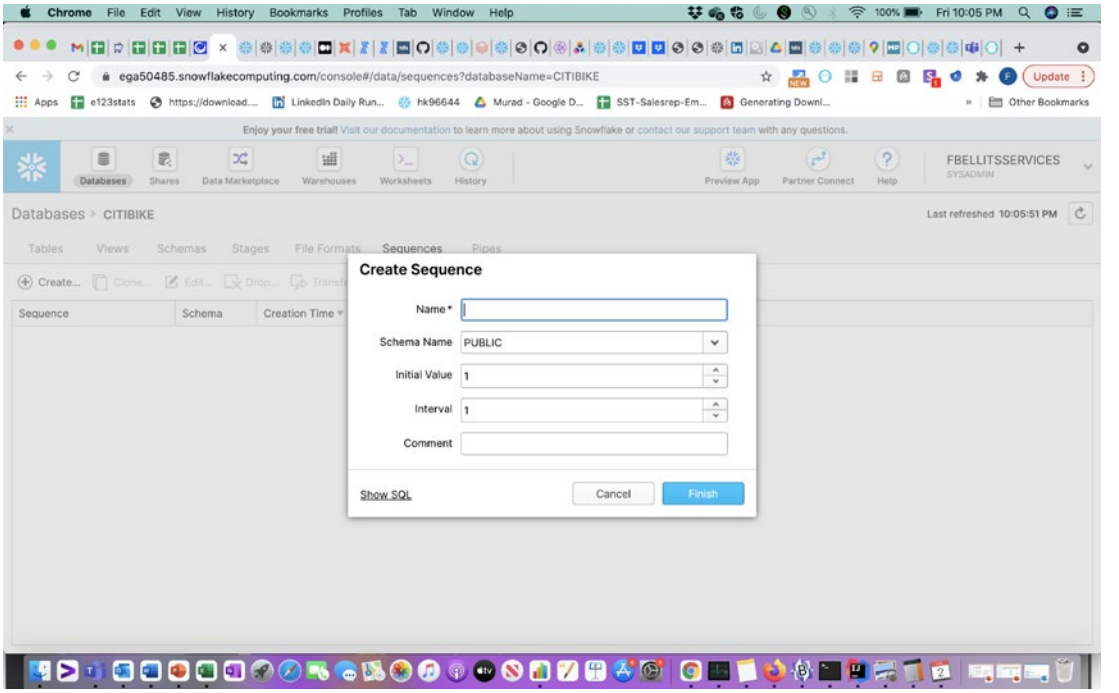

Figure 4-9. *Sequences*

You can Create, Clone, Edit, Drop, and Transfer Ownership of sequences within this web interface. You will notice once you select a sequence, similar to all the other database objects, you can grant permissions to the object as well on the right-hand side.

Pipes

Pipes or Snowpipes (the original name) within this interface are relatively new to the Classic Console. Pipes are one of the unique features of Snowflake that allow for the loading of continuous data pipelines as files are transferred into External Stages. Pipes can be set up to auto-ingest files into Snowflake based on the cloud provider event

mechanisms. Figure 4-10 shows the first Snowflake web interface form for creating an initial pipe. Similar to all Snowflake objects, you can also create a pipe with data definition code.

Tip Before you start creating your pipe, you will need a stage already created to use as the data source. Make sure to prepare the stage prior to creating the pipe. Also, if you plan to have a specific File Format you want to use in the pipe, then also create the File Format first before starting the pipe creation.

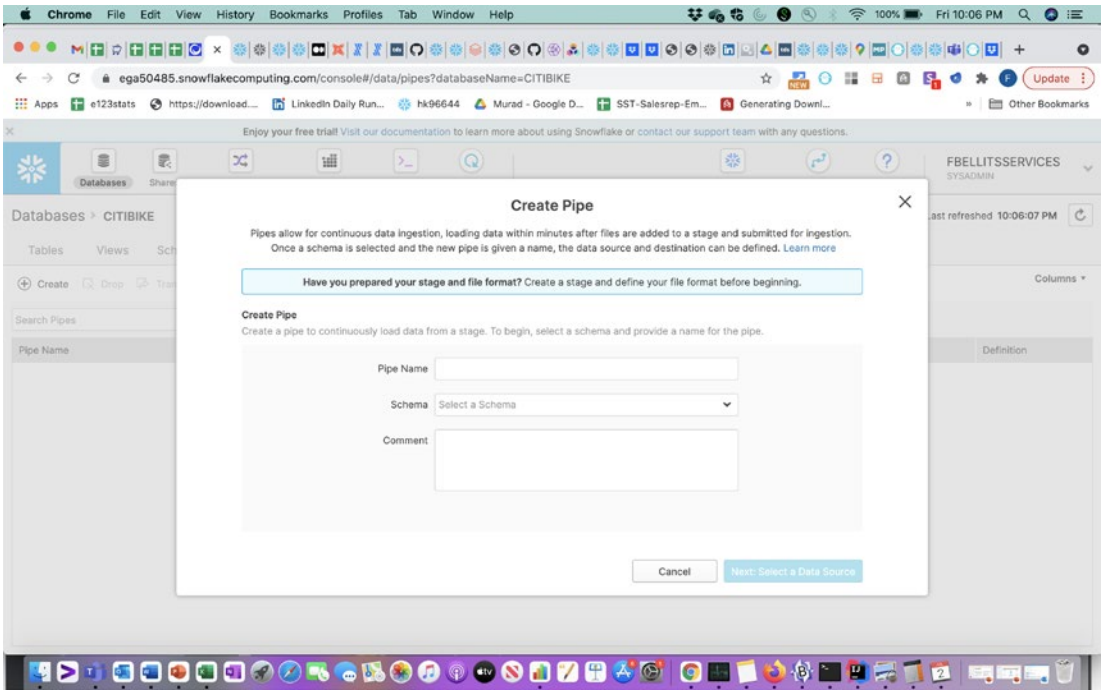

Figure 4-10. *Pipes*

Figure 4-10 shows the first of three dialogs that you fill out when creating a pipe. The three dialogs and the information you provide to each one are as follows:

Create Pipe: Screen 1

- Pipe Name: Enter the name you want to use for your pipe (Snowpipe).
- Schema: Enter the schema you want to create the pipe in.

- Comment: Enter a descriptive comment for what the pipe does.
- Click the button "Next: Select a Data Source" to go to the next Create Pipe screen.

Create Pipe: Screen 2 (Data Source for the Pipe)

- Stage: Select an existing stage name here from the dropdown. If you see "No options," you need to make sure you have access to a stage for the incoming data for the pipe.
- Enable Auto Ingest [this will ONLY be displayed if the stage has auto-ingest capabilities; otherwise, it will never show this checkbox]: This is a very key and important setting if you are looking to do automated ingestion. If this is checked, then you MUST set up the cloud provider file bucket correctly so that auto-ingest will work.
- File Formats (optional): Select an existing File Format here if you want the pipe to use that specific File Format. [This is optional and can already be specified in the stage itself as well.]
- Click the button "Next: Select the Data Destination" to go to the third Create Pipe screen.

Create Pipe Screen 3

- Data Destination: Select a schema and table where the pipe should copy the data into. The pipe works by executing a COPY INTO statement.
- Click the button "Create Pipe," and your pipe will be created.

Shares

Snowflake data shares are another unique and exciting feature that is part of the Snowflake Data Cloud. We will cover data shares, Data Exchanges, and Data Marketplaces in depth in Chapter 14. If you click the Shares icon and get the following message at the bottom of the screen as shown in Figure 4-11, then this means the current role you are using DOES NOT have access to create or view data shares:

Snowflake Data Sharing

Secure Shares enable you to consume data being shared with your organization and also provide data to others. Contact your account administrator to get access.

You need to switch to the ACCOUNTADMIN role, or a similar role provided by your organization, so you can view the screen in Figure 4-12, which shows a listing of data shares inbound. By default if you are using the ACCOUNTADMIN role, you will see two inbound data shares named ACCOUNT_USAGE and SAMPLE_DATA.

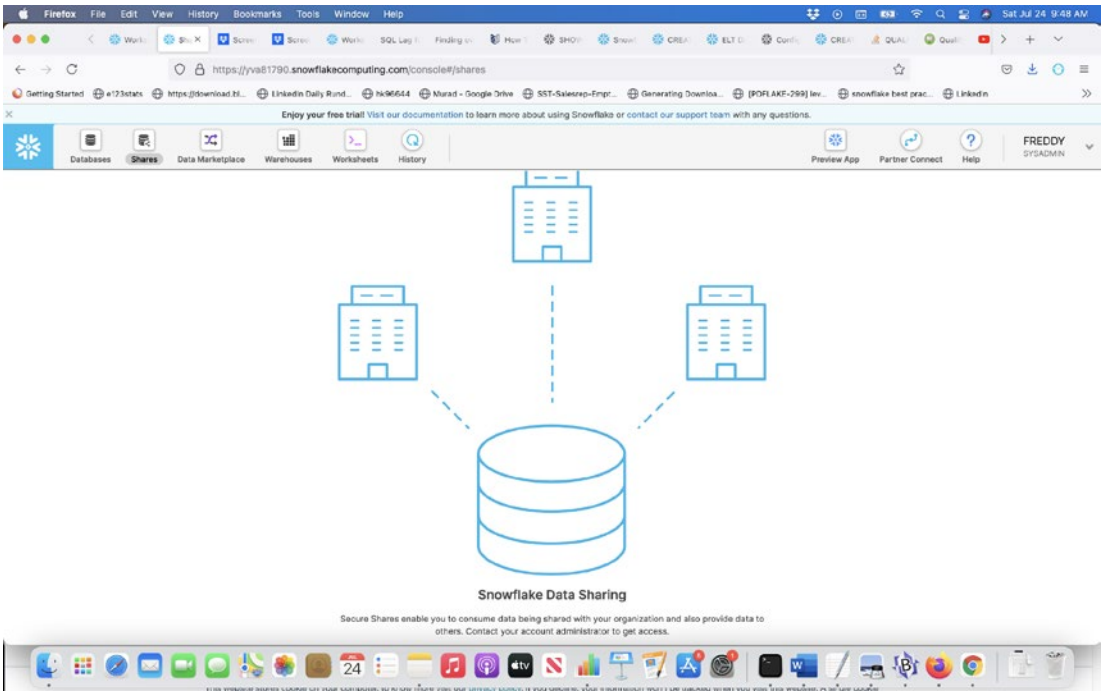

Figure 4-11. *Default View for Most Roles Besides ACCOUNTADMIN*

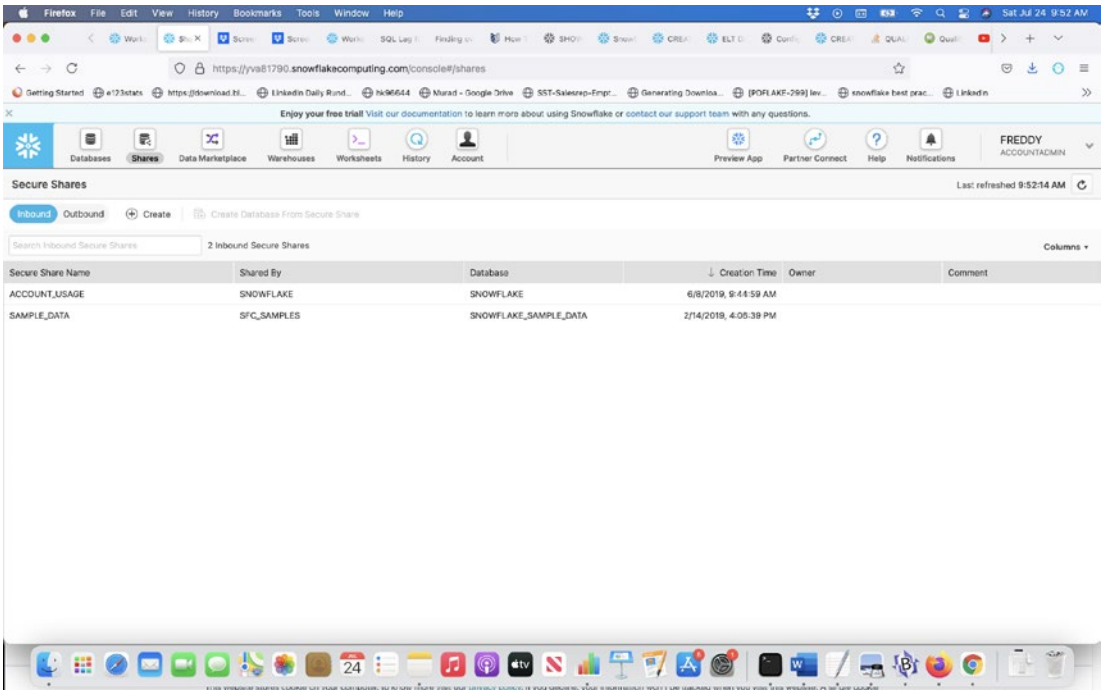

Figure 4-12. *Data Share Listings View for Roles with Data Share Access*

You will also notice in Figure 4-12 right next to the Inbound light-blue link is an Outbound link. If you click the Outbound link, you will see shares that you have created for external usage by other accounts. So let's dig into how you create an outbound or external data share to share between your own company, external suppliers, or any other external constituents you might have.

There are four parts to creating a data share:

Part 1: Before you can create a data share, you need to have the data properly prepared for sharing.

Part 2: Create the share itself.

Part 3: Review the secure share, preview tables, and validate secure views.

Part 4: Add consumers for the data share. This can be done by adding another account WITHIN that same region and giving it access to the share. If you try to add an account name not within the same exact region, you will get an error.

Figure 4-13 shows the first screen of creating a secure data share within the Snowflake Data Cloud.

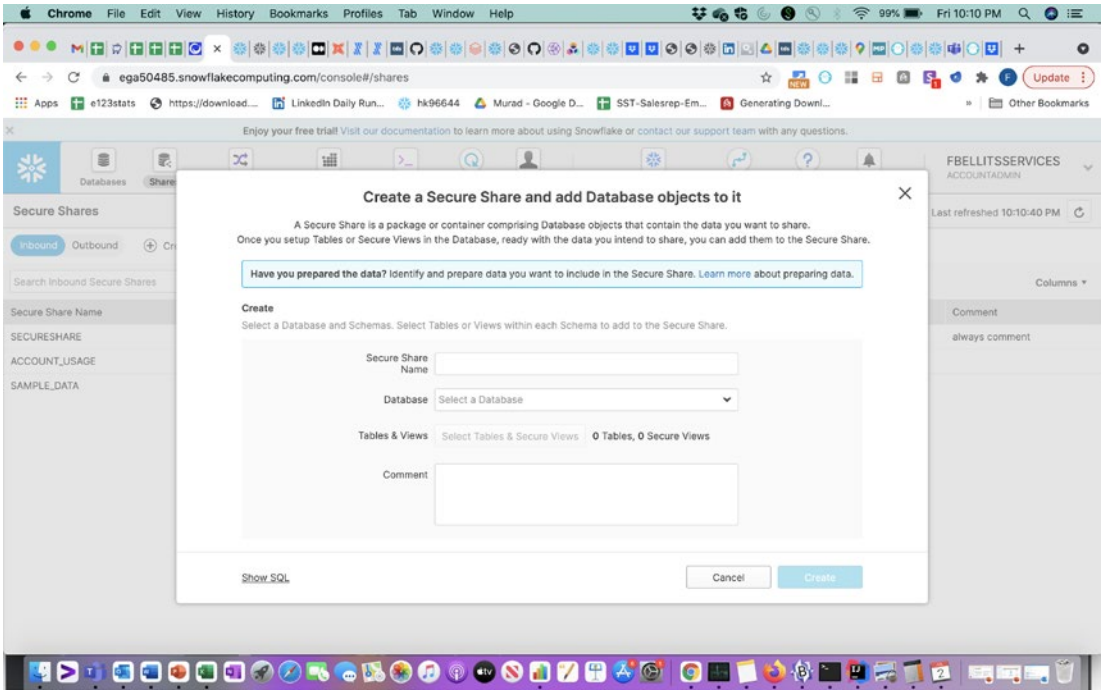

Figure 4-13. *Shares*

The form entries for a Snowflake data share are as follows:

- Secure Share Name: The name for the data share.
- Database: You select the database you plan to allow to be shared.
- Tables & Views: You select one or more tables and secure views (remember, regular views will not show up here).

You can add consumer accounts afterward.

Providing access to a share can be done either by adding existing accounts within the same exact region or by creating Reader Accounts. We will go into adding accounts in more depth in Chapter 14.

Data Marketplace

The Data Marketplace is one of the main reasons why Snowflake is much more than just a cloud database and is really the Snowflake Data Cloud. The Data Marketplace can be accessed through the Snowflake Classic Console by clicking the icon, but at this time you still need to reauthenticate because you are technically going to the Snowsight Preview App interface where the Data Marketplace is hosted. Figure 4-14 shows what the Data Marketplace Classic Console splash screen looks like now.

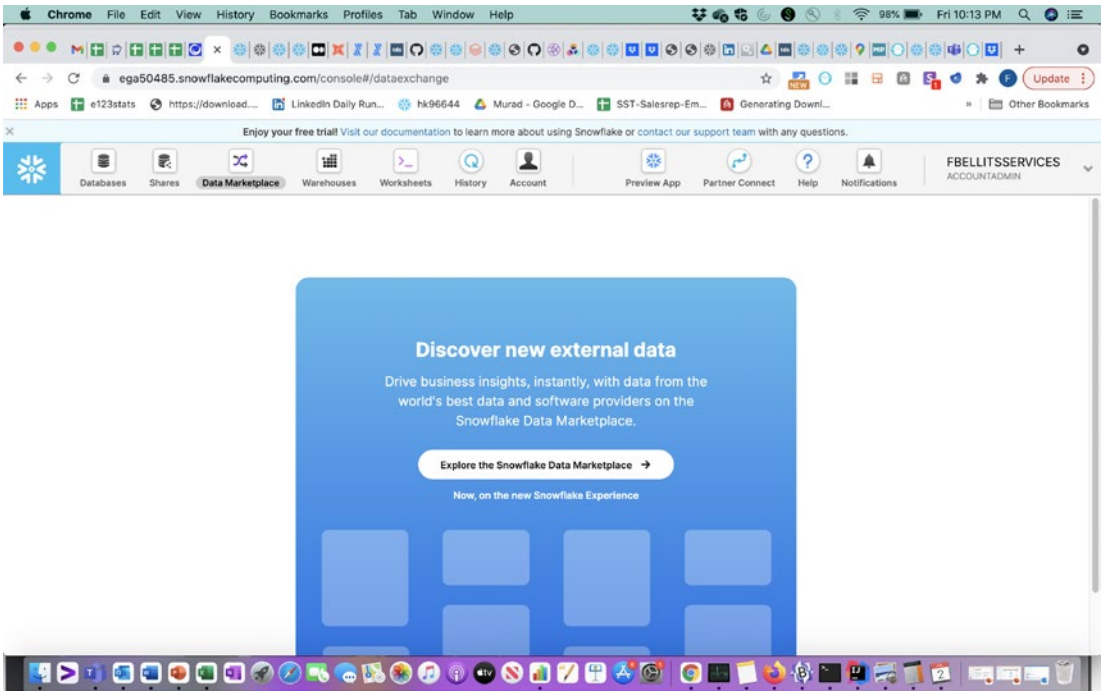

Figure 4-14. *Data Marketplace Splash Screen*

Once you click the "Explore the Snowflake Data Marketplace" button in Figure 4-14 and authenticate to the Snowflake Preview App, then you will be presented with the initial Data Marketplace Dashboard, which has listings of hundreds of Data Marketplace shares from hundreds of data providers. Figure 4-15 shows an example of the Data Marketplace initial interface. Also, remember that the Data Marketplace can be different from region to region right now. We also discuss the Snowflake Data Marketplace in depth in Chapter 14.

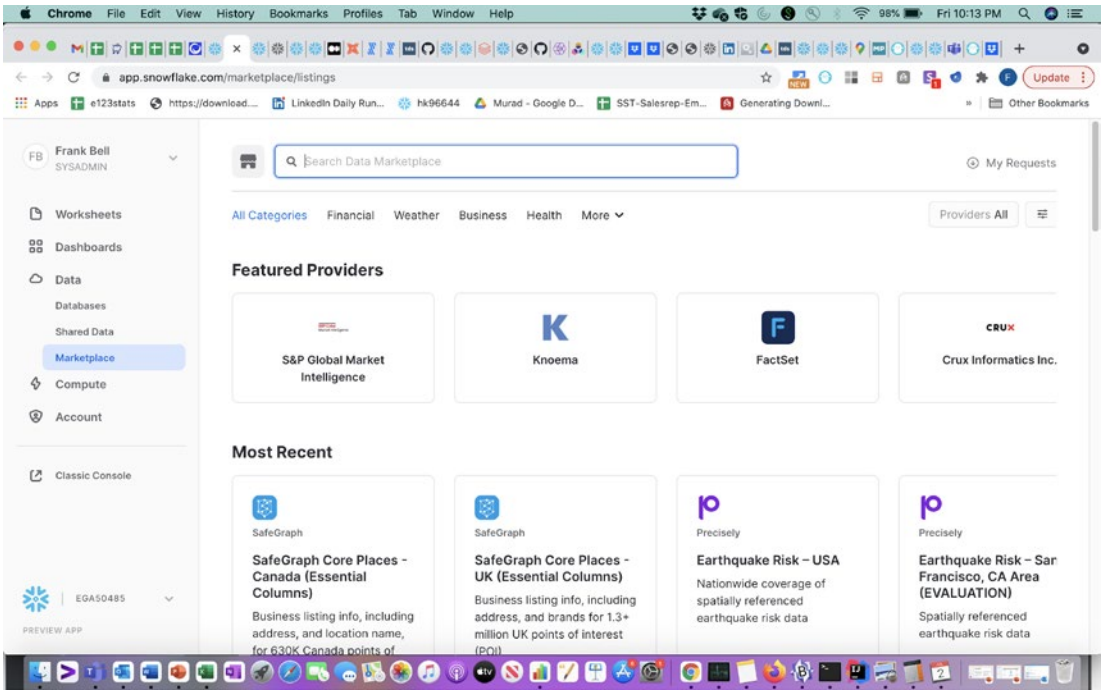

Figure 4-15. *Data Marketplace Listings Initial Screen*

In Figure 4-15 you can search for Data Marketplace shares based on provider name or share name or choose one of the Data Marketplace categories.

Warehouses (Named "Compute Warehouses" on the Preview App)

Warehouses are a new concept within the Snowflake Data Cloud, and they are technically virtual warehouses or really just pointers to compute resources within Snowflake and the cloud provider on top of which you are running Snowflake. You cannot execute SELECT, DELETE, UPDATE, INSERT, and MERGE statements without a warehouse assigned. If you are coming from a standard relational data warehousing background, then do NOT get confused by the naming convention Snowflake used here with "warehouses." Snowflake warehouses are completely virtual pointers and have nothing whatsoever to do with the storage of the data. The Snowflake Data Cloud has a clear separation of storage from compute, and warehouses are the compute part of the Snowflake Data Cloud.

Figure 4-16 shows the Create Warehouse form in the Classic Console on Snowflake.

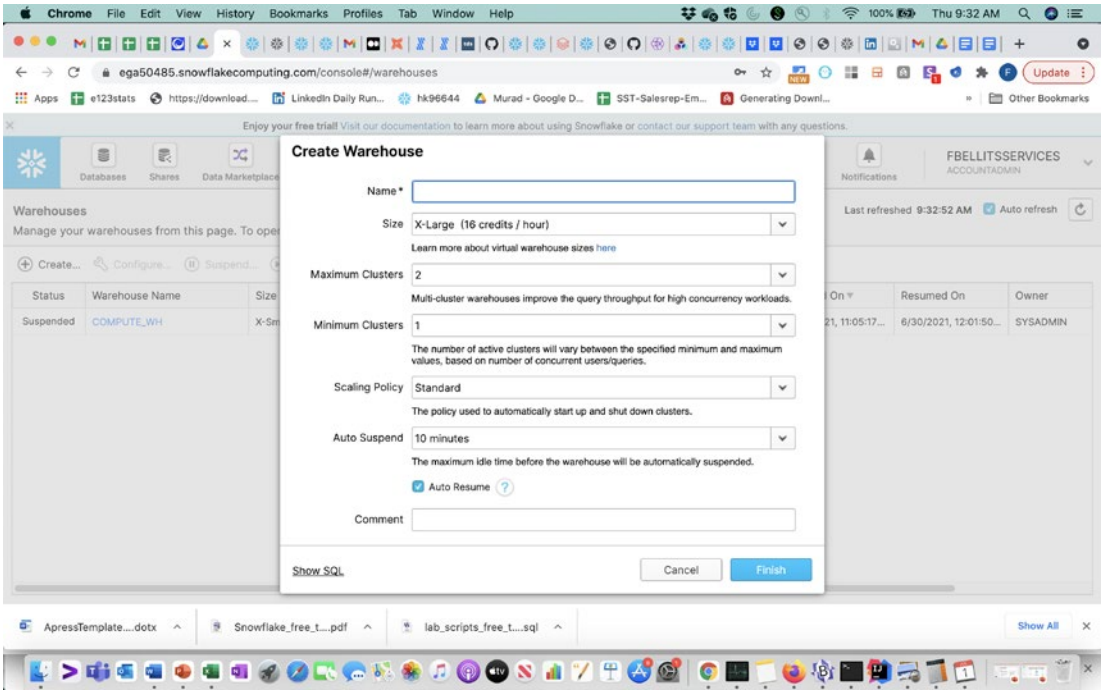

Figure 4-16. *Warehouses*

The form entries required for the Snowflake warehouse are

Name: The name of the warehouse.

Size: This is one of the most important selections and has MASSIVE impacts on the cost of your Snowflake Data Cloud. The cost difference between the smallest warehouse of XS (extrasmall) and the largest of 6XL is significant.

Maximum Clusters: (You will not see this line if you have the Snowflake Standard version). This value can range from 1 to 10 currently.

Minimum Clusters: (You will not see this line if you have the Snowflake Standard version). This can range from 1 to 10 currently.

Scaling Policy: Economy or Standard. This impacts how fast an additional cluster turns on.

Auto Suspend: This is another incredibly important setting related to both cost and performance.

Comment: It is important to add your comment on what this warehouse is created for.

Worksheets

If you use the web interface exclusively, then this will be your main working area. The most important part of the worksheets in both the Classic Console and the Preview App is the context selection. This greatly impacts what your worksheet syntax must be. If you are running a command that is not fully qualified with the DatabaseName. SchemaName.FinalObjectName, then Snowflake will use what you have defined within the context for your database and schema. Also, unless you change the warehouse setting in the context, then Snowflake will also use that warehouse to run the worksheet workload details and charge credits based on the warehouse size and other settings. The warehouse will resume and suspend based on the warehouse Auto Suspend and Auto Resume settings as well, which have significant impacts on usage costs. Figure 4-17 shows a blank worksheet tab with the context dropdown settings shown.

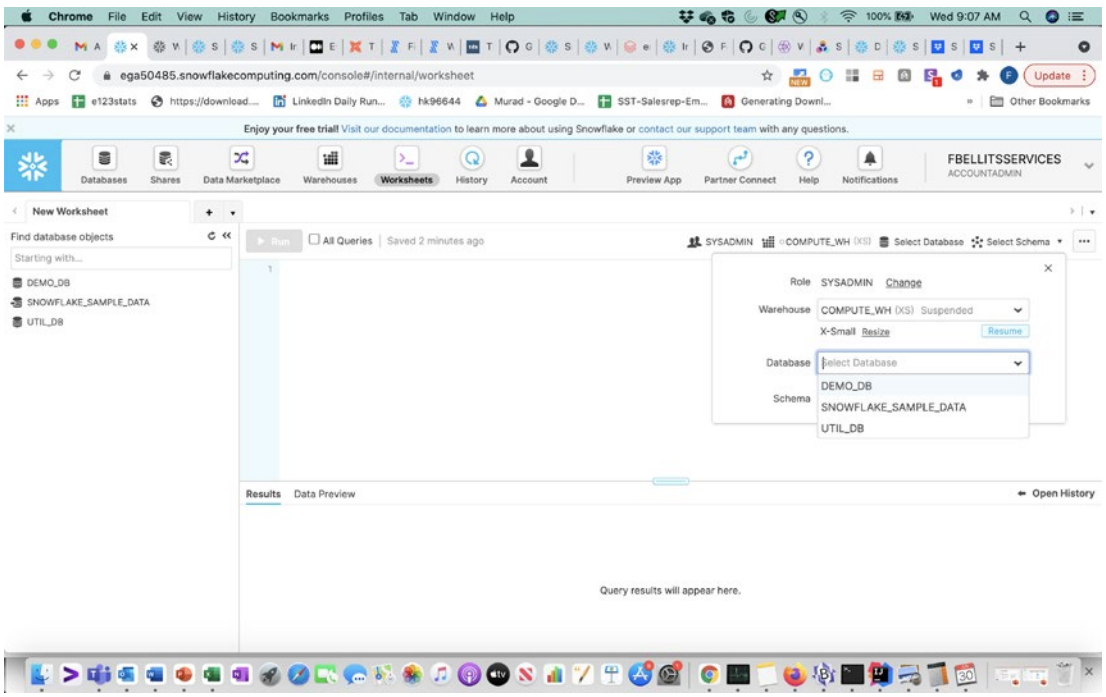

Figure 4-17. *Worksheets*

You will also notice on the Classic Console worksheet that the left navigation tree has both object exploration and search functionality. You can view the hierarchy of databases, schemas, and objects you have access to within the navigation tree. Remember that the navigation tree is dependent upon the current worksheet context that you are using. The worksheet has four main areas including the navigation tree, the worksheet input itself, the worksheet context (where you select Role, Warehouse, Database, and Schema), and the results pane. The results pane also has many options to view the query history or to copy or export data from the results. You can also access query profiles from here as well.

Tip The role you are using when you create an object in the worksheet has a large impact on what roles and users can view or use the object. It is incredibly easy to mistakenly create an object (table, view, schema, and even database) in the worksheet while being in the role of ACCOUNTADMIN and forget to grant ownership or visibility to another role that needs access. When someone states they cannot find some object that you know has been created, the first thing to troubleshoot is what roles have access to the object.

History

The History icon is one of my favorite initial features within the Snowflake Data Cloud. Many preexisting relational databases had history logs of every action that was taken, but the Snowflake Data Cloud database really was the first database I came across that was so transparent about providing full history of every action that took place easily (assuming you have the privileges to see the actions). Snowflake does control access to the query history so that only the users who have the proper role access can actually view the queries they have access to in the history log. This is also one way you can access query profiles related to any query run on your account that you have access to. Figure 4-18 shows an example history log on the Classic Console web interface. You can select a query profile by clicking the query ID within this interface. Query history is available for up to 14 days. If you need further query history, then you can query the Snowflake internal view named SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY.

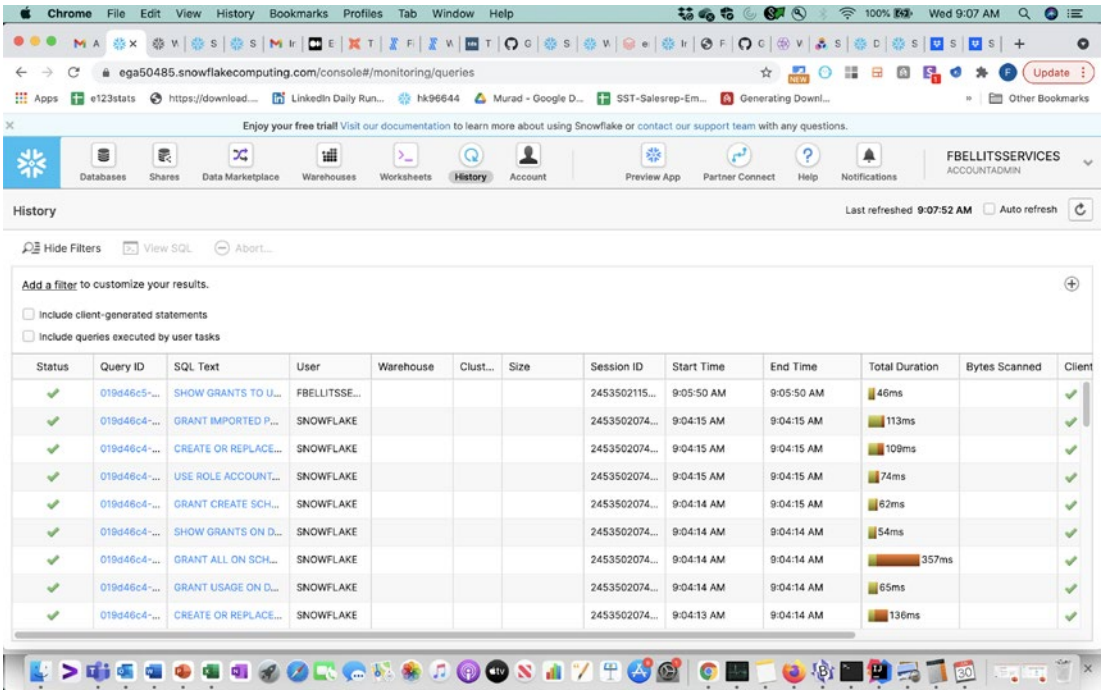

Figure 4-18. *History (Very Cool)*

Partner Connect

Partner Connect is another unique concept Snowflake came up with to showcase approved partner functionality and easily trial partner solutions. Figure 4-19 shows the initial top of the screen that you will see on the Partner Connect web interface. You can scroll down to find additional partners as well.

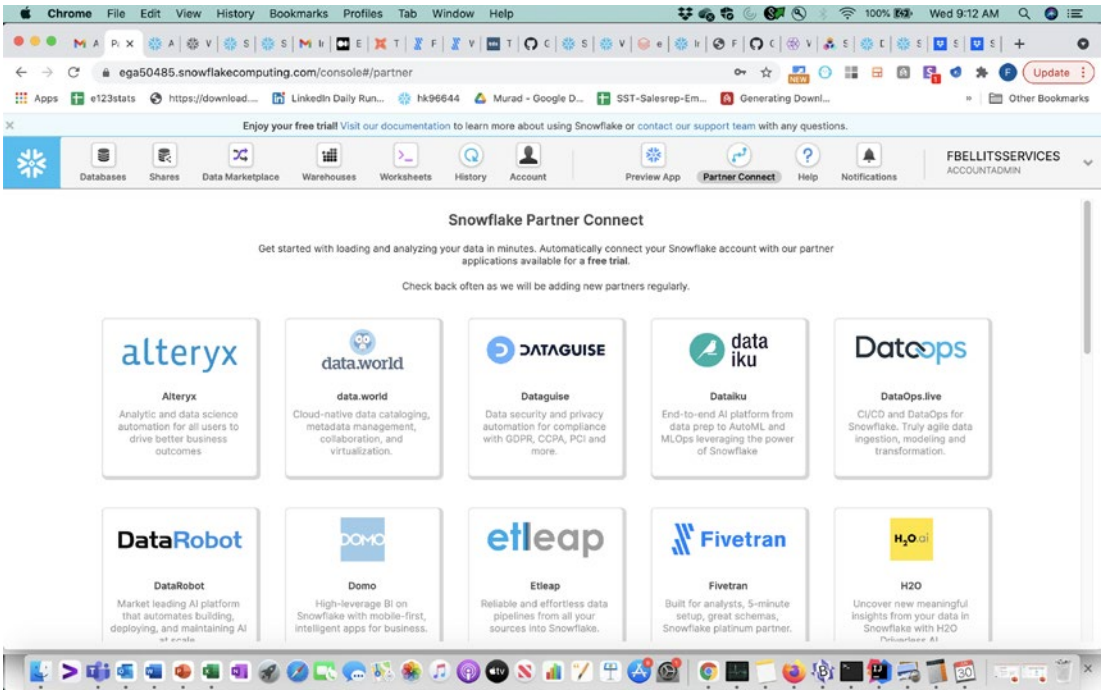

Figure 4-19. *Partner Connect*

Help

Help is a key component of any technical system. Snowflake has a simple but useful selection of help links to four main areas including the Snowflake documentation, Snowflake Community, Downloads, and the Help Panel. Most of these are straightforward and make it easy to find help details on Snowflake. The Downloads section provides links and installation details on how to set up Snowflake connectors and tools including the CLI client, JDBC driver, ODBC driver, Python components, Node.js driver, Spark connector, Go Snowflake Driver, and SnowCD (the Snowflake Connectivity Diagnostic Tool). Figure 4-20 shows the top-level options available when you click the Snowflake Classic Console Help icon.

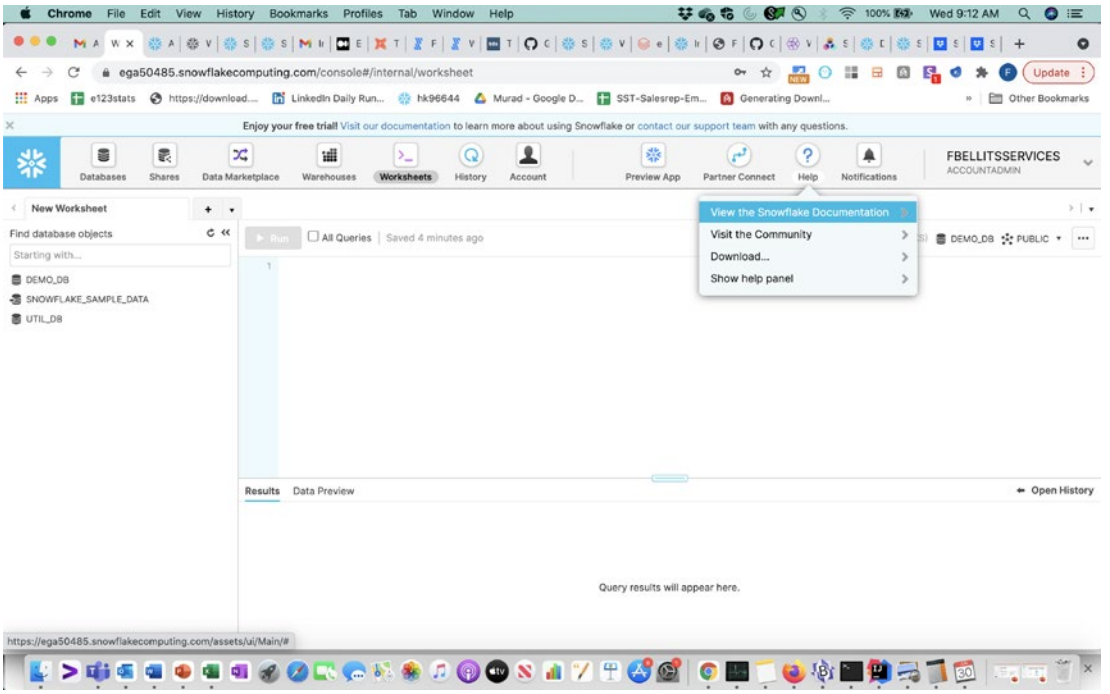

Figure 4-20. *Help Dropdown Screen*

Notifications

Notifications ONLY provide notifications to ACCOUNTADMINS currently. These notifications are important for monitoring spend on virtual warehouses and are related to resource monitors. They should be one of the first things that are set up at the beginning of gaining access to your Snowflake Data Cloud system. I recommend setting up BOTH web and email notifications at first and immediately configuring resource monitors, which we will discuss later. Figure 4-21 shows the view when you hover on the Notifications icon. You must click the settings link in Figure 4-21 to get to Figure 4-22 where you can set the notification options.

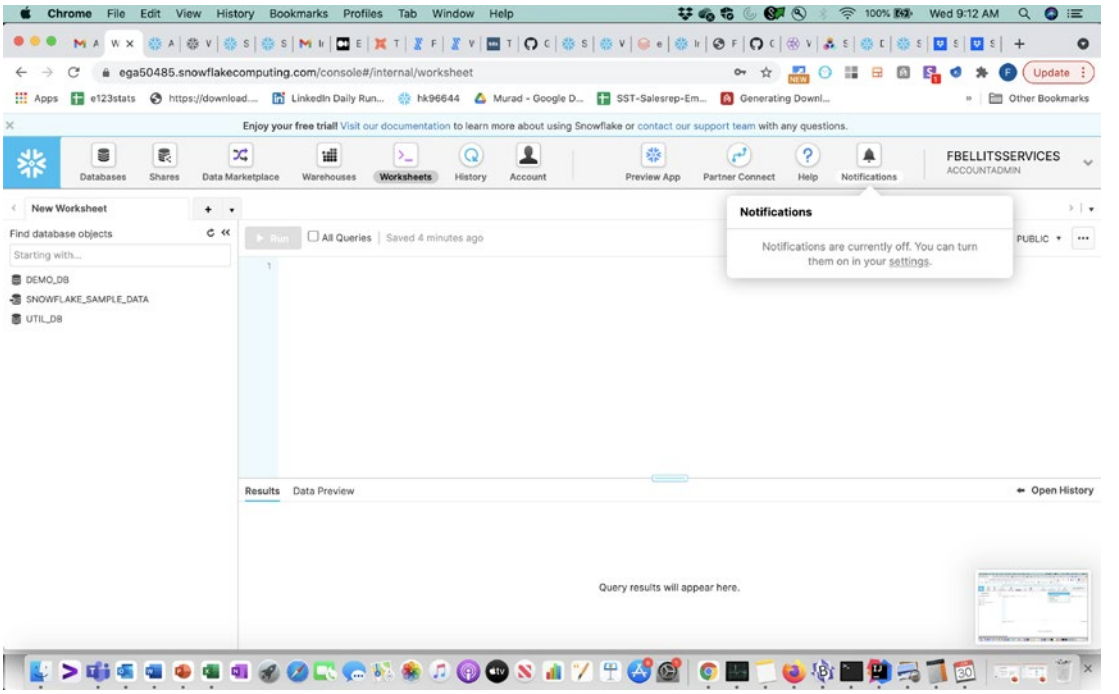

Figure 4-21. *Initial Notifications Icon View*

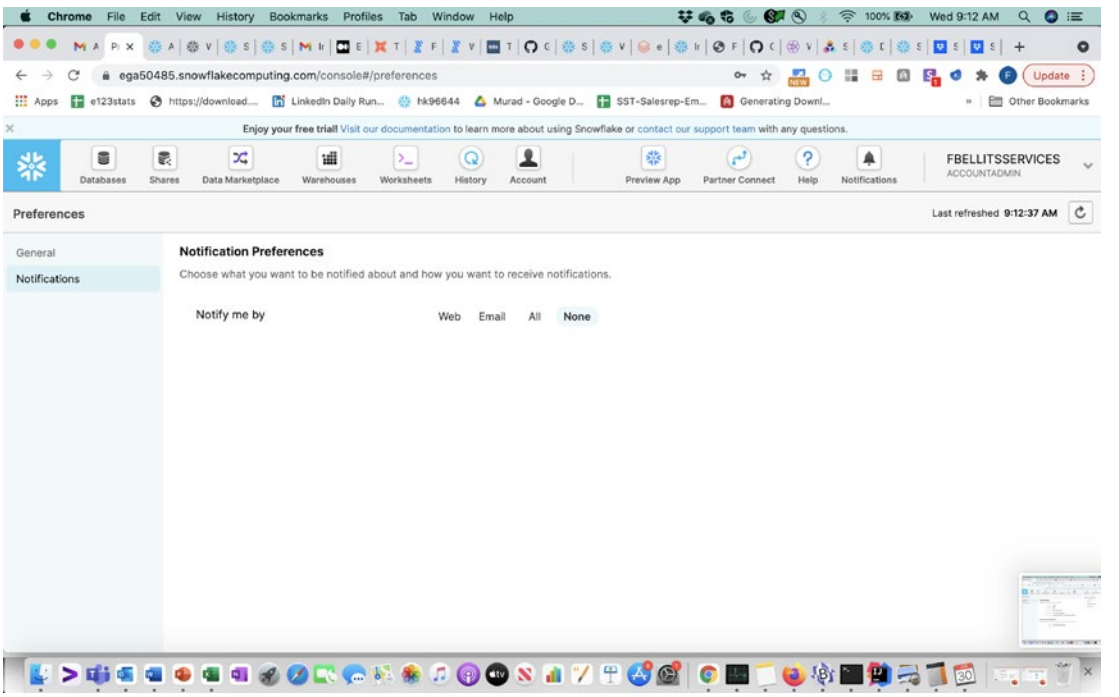

Figure 4-22. *Classic Console – Notification Preferences Screen*

Account

When you click the Account icon, you are taken to a web page with eight standard options including Usage, Billing, Users, Roles, Policies, Sessions, Resource Monitors, and Reader Accounts.

Tip If you do not see the Account icon to the right of the History icon on the Classic Console, then you do not have rights to access the Account details with the current role you are in. Try changing the role to ACCOUNTADMIN if you have that access.

Usage

The Usage screen allows users to see their current daily compute spend and the spend breakdown by warehouses if the first rectangle with Warehouses and Credits Used is selected. Figure 4-23 shows what this warehouse spend view looks like on a new Snowflake Data Cloud account.

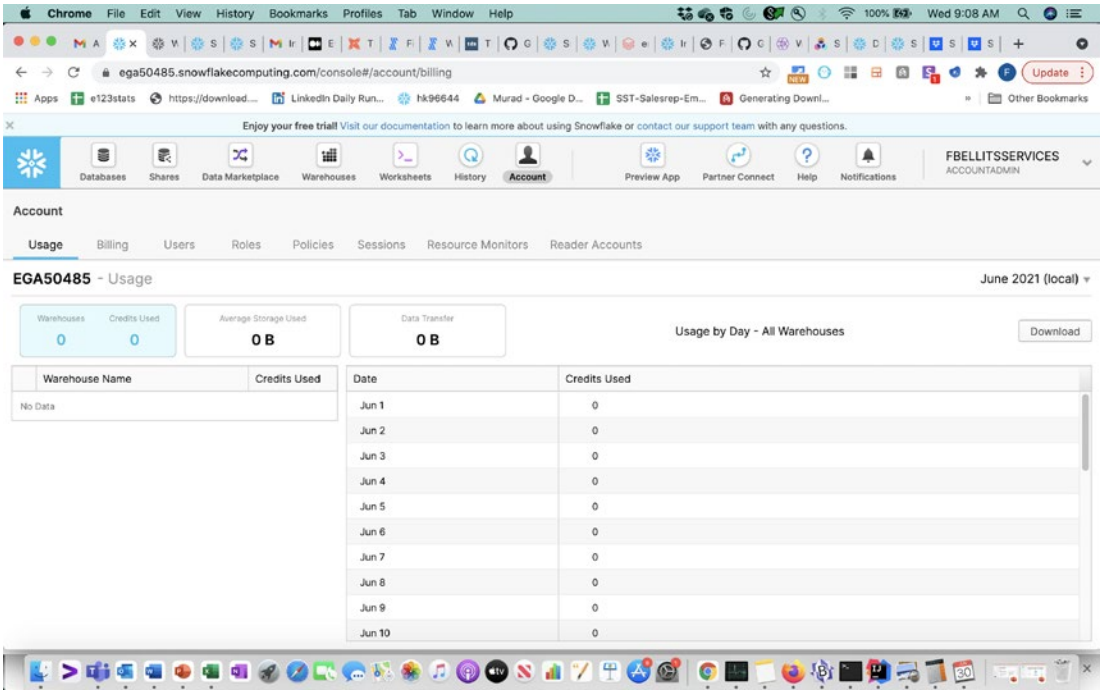

Figure 4-23. *Usage*

If you want to view storage usage, then you would click the rectangle just to the right of the "Warehouses and Credits Used" one, which is labeled "Average Storage Used." This view breaks down storage by database, stage, and Fail Safe. Finally, you can view any data transfer costs by clicking the rectangle to the right of this named "Data Transfer."

Billing

The Billing screen is where you can view and manage billing information for your account and add your credit card. This is typically used by on-demand accounts. Figure 4-24 displays an example of the Billing screen.

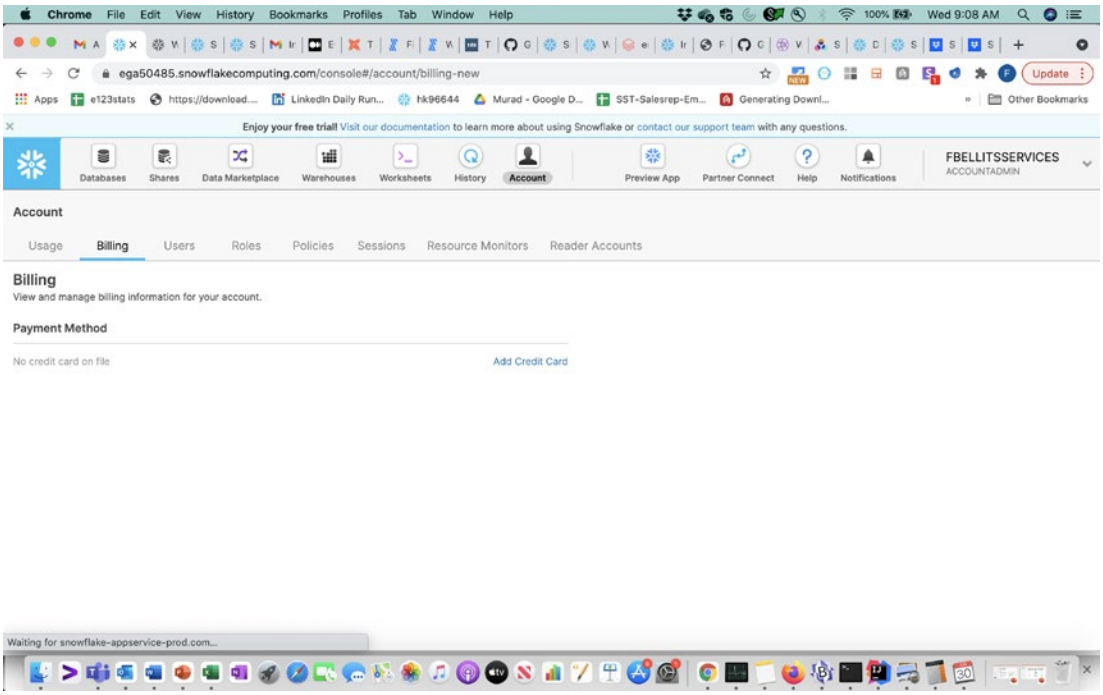

Figure 4-24. *Billing*

Roles

Roles are a key part of how data is secured and accessed on the Snowflake Data Cloud. Users can only access functionality and data based on roles associated with them. Figure 4-25 shows the listing of standard roles that are set up initially on the Snowflake Data Cloud including ACCOUNTADMIN, ORGADMIN, PUBLIC, SECURITYADMIN, SYSADMIN, and USERADMIN.

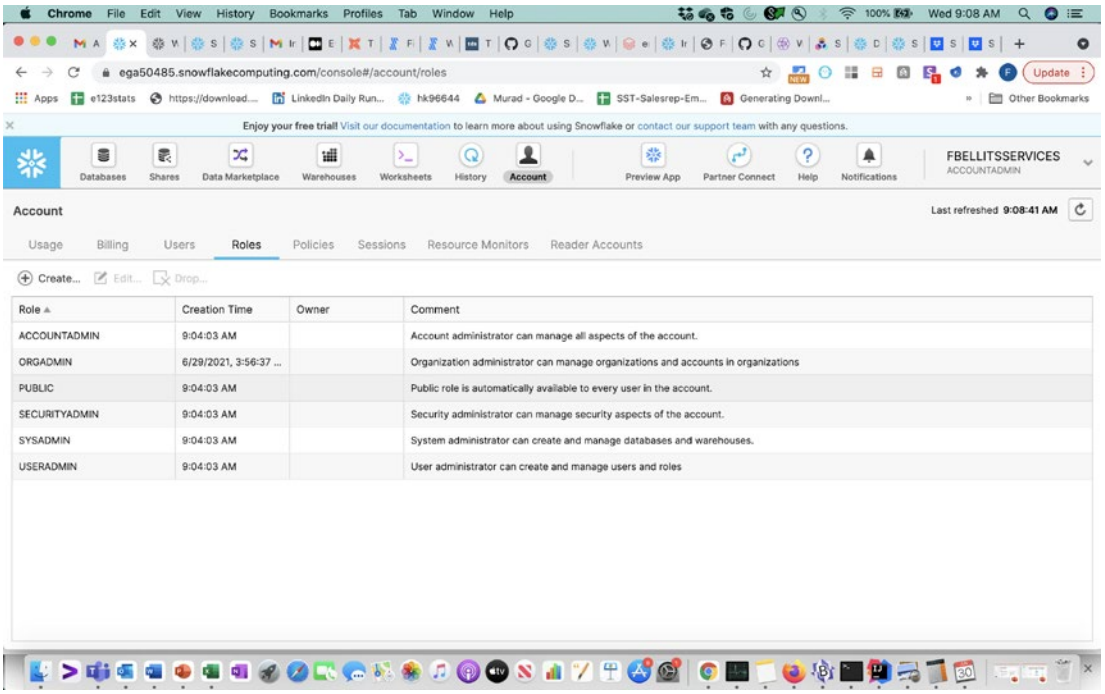

Figure 4-25. *Roles*

Policies

Policies in Snowflake are used for network security. Figure 4-26 shows an example of the Create Network Policy form. You can choose what IP addresses you want to allow and those you want to block. You can have multiple policies as well.

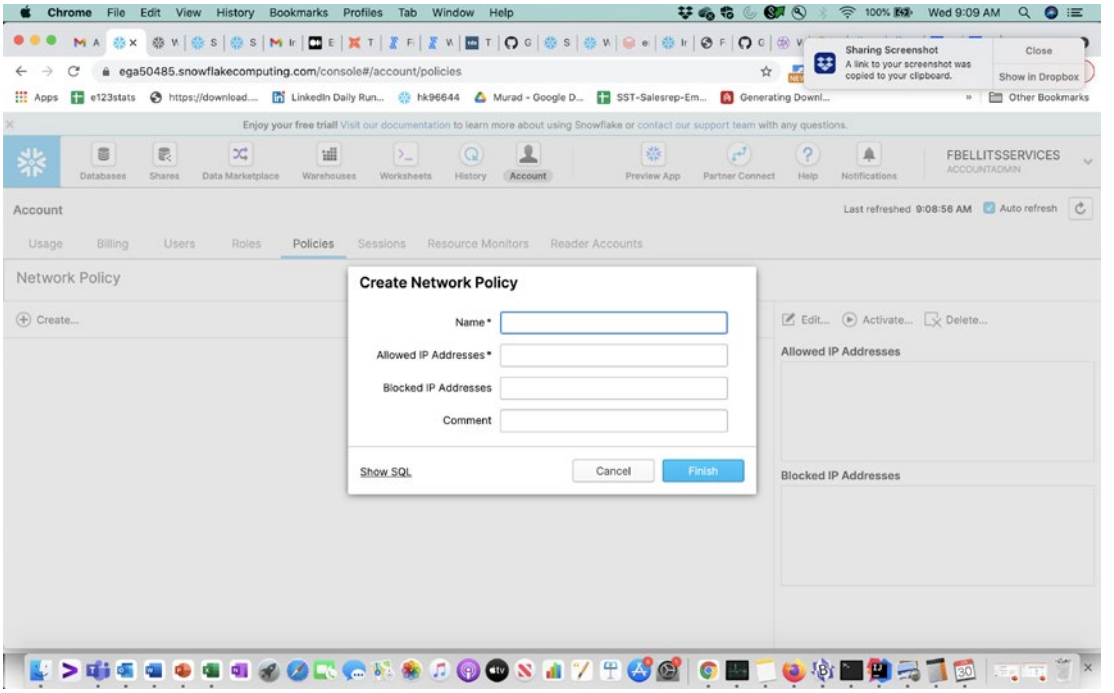

Figure 4-26. *Policies*

Sessions

The Sessions link under the Account area allows you to see currently active sessions in your Snowflake Data Cloud account. The session listings include details on User Name, Session ID, Open, Start Time, End Time, Duration, Expiration Time, Client Driver, Client Net Address, and Authentication Method. Figure 4-27 shows the session display with two active sessions.

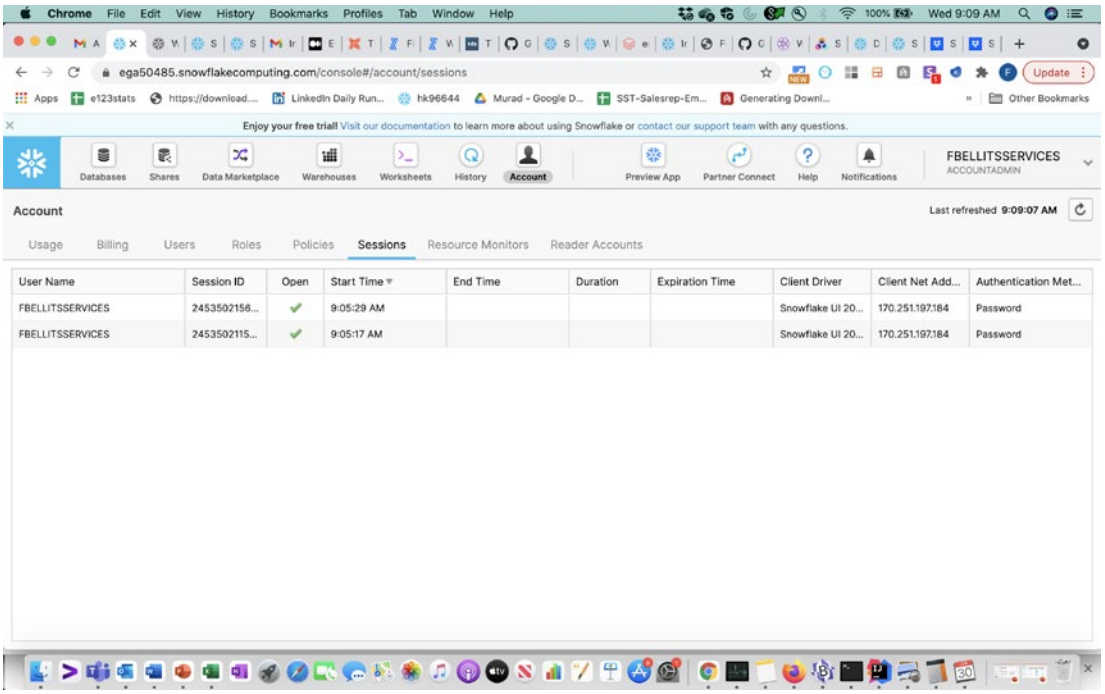

Figure 4-27. *Sessions*

Resource Monitors

Resource monitors are one of the most important and often neglected features of the Snowflake Data Cloud. When on-prem users transition to the Snowflake Data Cloud, they often love the flexibility, ease of use, and unlimited scalability. Database practitioners who used on-prem databases must make a shift to using tools to monitor cloud consumption, storage, and other costs. Most on-prem databases just had fixed prices for their systems, but cloud databases typically have consumption pricing. The ONLY tool currently available by default on the Snowflake Data Cloud that monitors consumption and shuts off warehouses are resource monitors.

Resource monitors can be used in two main ways to either notify Account Admins of hitting usage thresholds or to shut down the warehouse. One or multiple notifications can be set up. When a warehouse hits resource monitor thresholds, then it can either be suspended immediately or after the current query that went over the threshold finishes.

Figure 4-28 is an example of the Create Resource Monitor form. You will notice you can set up resource monitors for one warehouse, multiple warehouses, and overall accounts. Also, warehouses currently can be set for monitoring intervals of Daily, Weekly, or Monthly.

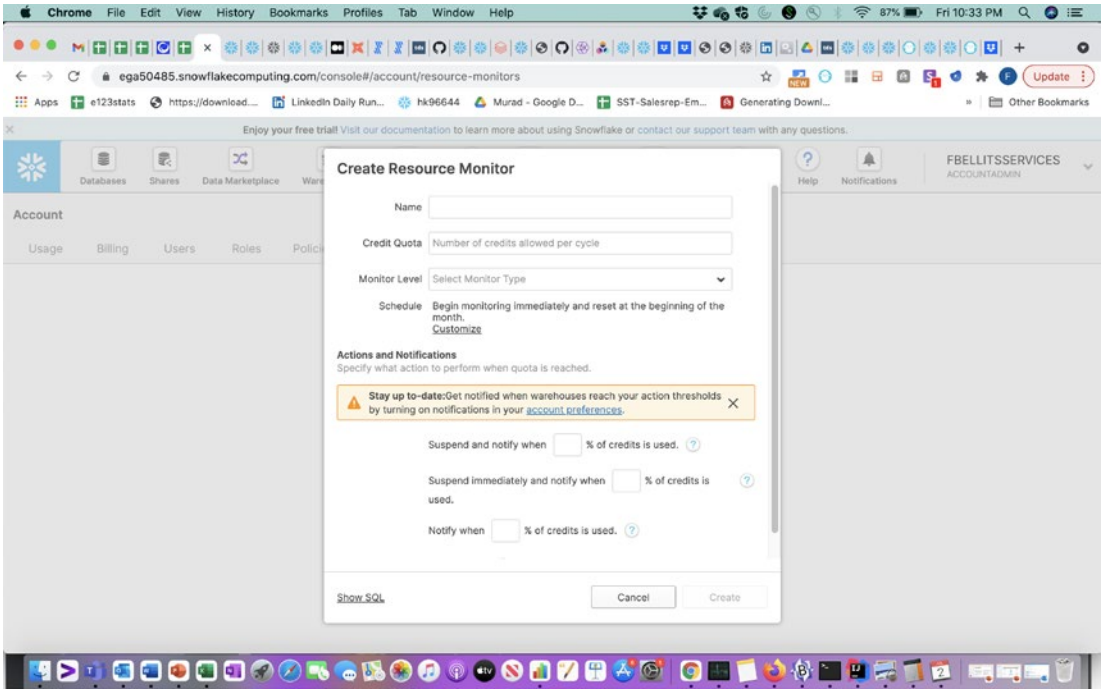

Figure 4-28. *Resource Monitors*

Reader Accounts

Reader Accounts are a type of account Snowflake created to allow data share providers a way to share their data with data consumers who do not have an existing Snowflake account. The main difference between a Reader Account and a normal Snowflake account is that the Reader Account consumption is paid for by the data provider. Also, the data provider is the one who creates the Reader Account. Figure 4-29 shows the standard form where you can create a Reader Account assuming you are currently using a role like ACCOUNTADMIN.

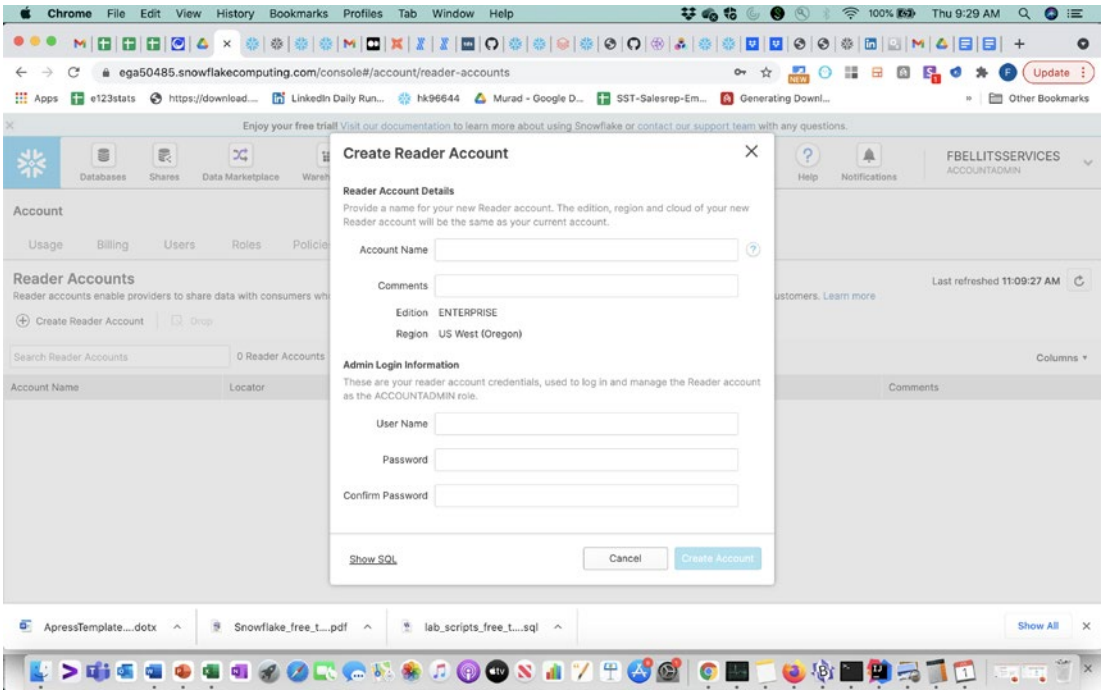

Figure 4-29. *Create Reader Accounts*

Reader Accounts make it incredibly easy for a data provider to share data to their customers who do not have an existing Snowflake account. You can easily set them up and share data within minutes. While they provide great ease of use and power, you must remember these are full Snowflake accounts that the primary account is paying the Snowflake costs for. You should never provide the administrative username and password to the reader party unless you have some agreement in place for the costs and administrative responsibilities. Otherwise, the best practice is to go in as the administrator username you just created and create roles with typically limited access specially to creating and using warehouses. We also recommend setting up resource monitors immediately so you can track all the usage effectively on the Reader Account.

The Create Reader Account form on the Snowflake Classic Console shows the fields you need to fill out to create a Reader Account. The following are the field names and their descriptions:

- Account Name: Provide an account name for the new Reader Account that you [the primary account] will be paying the compute and storage for.
- Comments: Fill in details of what the Reader Account will be used for.
- Edition [this is set to what your current account is]
- Region [this is set to what your current account is]
- Admin Login Information: Create an administrative username and password for the Reader Account.
- Click the button "Create Account" to create the new Reader Account. Remember that Snowflake accounts now typically take a minute to get fully created. If you go to the account URL too early, you may see a 400 error.

Once you are finished creating the Reader Account, make sure to set up the security as mentioned previously before giving access to the reader data consumers. Remember that all the usage performed on the Reader Account will be paid for by the primary account, which created the Reader Account. In Figure 4-30 you can see what the initial screen will look like for the new Reader Account.

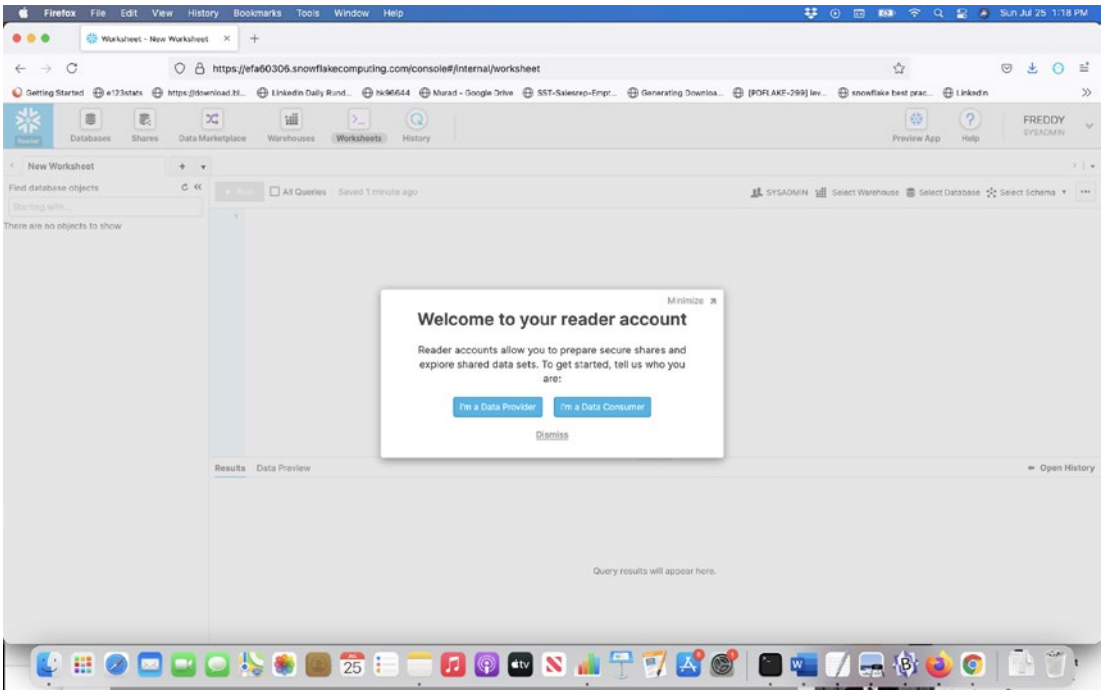

Figure 4-30. *New Reader Account*

Summary

The Snowflake Classic Console has been the standard web interface for using the Snowflake database since it launched. It still provides an excellent initial interface for executing queries on Snowflake and also managing the Snowflake Data Cloud. The new Snowsight interface, which we cover in Chapter 5, does provide some amazing new features related to autosuggestion, autocompletion, and sharing of worksheets and dashboards. Currently you also must access Snowsight (Preview App) through the Snowflake Classic Console. There is no current date scheduled for removing the Classic Console, so all of us veterans used to using it can continue to do so for a while.