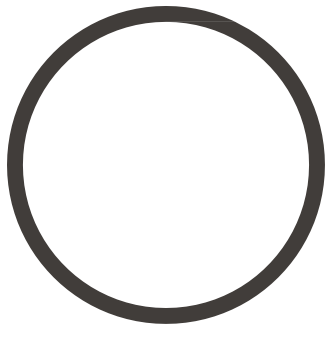

Data Engineering SNOWFLAKE

ALL CONCEPTS TO GET STARTED

Cloud Data Warehouse

A cloud-based platform for storing and analyzing data, which offers scalability, flexibility, and cost-efficiency compared to traditional on-premises data warehouses.

Snowflake provides a fully managed service with separate compute, storage, and cloud services layers, making it easier to scale and manage data operations.

Snowflake

Architecture

Snowflake's architecture separates storage and compute, allowing for independent scaling and efficient data management. This design eliminates many limitations of traditional data warehouses.

Snowflake uses a multi-cluster shared data architecture, where storage is centralized, and compute resources can be scaled up or down independently based on workload.

Virtual Warehouse

A virtual warehouse is a cluster of compute resources in Snowflake. Each virtual warehouse can be scaled independently to match the workload, providing the necessary compute power for query execution without affecting other warehouses.

If a company needs to run a heavy analytical query during peak business hours, they can scale up the virtual warehouse to a larger size, ensuring faster query performance. After the peak hours, the warehouse can be scaled down to save costs.

Database

A logical grouping of schemas, tables, and other database objects. It provides a namespace for organizing and managing data.

Creating a new database in Snowflake: *CREATE DATABASE sales_data;*.
This command sets up a new database where all sales-related schemas and tables can be organized.

Schema

A logical grouping of database objects such as tables, views, and stored procedures. Schemas help organize objects within a database.

Creating a new schema in a database: *CREATE SCHEMA sales_data.january;*. This schema can contain all tables related to January's sales data.

Table

A structured set of data elements (values) organized in rows and columns. Tables are fundamental storage objects in a database.

Creating a new table:
CREATE TABLE customers
(id INT, name STRING, email STRING);.

This table stores customer information.

View

A virtual table based on the result-set of a SQL query. Views do not store data themselves but provide a way to represent data stored in tables.

Creating a new view:
CREATE VIEW vip_customers
*AS SELECT *FROM customers*
WHERE status ='VIP';.
This view shows only VIP customers.

Stage

A location where data files are stored temporarily before being loaded into Snowflake tables. Stages can be internal or external (e.g., S3, Azure Blob Storage).

Creating an internal stage: *CREATE STAGE my_stage;*.
This stage can be used to store data files before loading them into tables.

File Format

Defines the format of data files to be loaded into Snowflake (e.g., CSV, JSON, Avro). File formats specify how Snowflake should interpret the contents of the files.

Creating a file format for CSV files:
CREATE FILE FORMAT my_csv_format TYPE ='CSV' FIELD_OPTIONALLY_ENCLOSED_BY ="";.

Warehouse Size

Snowflake offers different sizes for virtual warehouses (e.g., X-Small, Small, Medium, Large) to accommodate various workloads. Larger sizes provide more compute resources.

A Small warehouse might be sufficient for routine queries, while a Large warehouse can handle complex analytical queries. Adjust the size based on workload demands.

Scaling Up

Increasing the size of a virtual warehouse to provide more compute resources for a specific workload.

Scaling up a virtual warehouse:
ALTER WAREHOUSE my_warehouse SET WAREHOUSE_SIZE ='LARGE';.
This increases the compute power available for queries.

Scaling Out

Adding more compute clusters to a virtual warehouse to handle increased concurrency and workload demands.

Enabling auto-scaling for a warehouse:
ALTER WAREHOUSE my_warehouse SET MAX_CLUSTER_COUNT =5;.
Snowflake will add clusters as needed to handle concurrent queries.

Auto-Suspend

Automatically suspends a virtual warehouse when it is idle for a specified period, saving costs.

Setting auto-suspend for a warehouse:
ALTER WAREHOUSE my_warehouse SET AUTO_SUSPEND =300;.
The warehouse will suspend after 5 minutes of inactivity.

Auto-Resume

Automatically resumes a suspended virtual warehouse when a query is submitted, ensuring availability without manual intervention.

Enabling auto-resume for a warehouse:
ALTER WAREHOUSE my_warehouse SET AUTO_RESUME =TRUE;.
The warehouse will resume automatically when a query is submitted.

Query Caching

Snowflake caches the results of queries to speed up repeated query executions, reducing the need for re-computation and saving compute resources.

Running the same query twice will utilize the cached result if the underlying data has not changed, improving performance and efficiency.

Result Cache

Stores the results of queries executed within the past 24 hours. The cache is accessible to all users within the account, reducing compute costs and speeding up query performance.

If a query is run and then re-run within 24 hours without changes to the underlying data, the result is fetched from the result cache, saving compute resources.

Metadata Cache

Stores metadata about database objects to speed up query parsing and planning. This cache helps optimize query execution by reducing the time needed to access metadata.

Metadata about tables, columns, and statistics is cached, allowing faster query planning and execution. This helps Snowflake optimize performance for complex queries.

Data Caching

Snowflake caches data in the local storage of virtual warehouses to improve query performance. This cache is independent for each virtual warehouse.

Frequently accessed data is stored in the local disk cache of a virtual warehouse, reducing the need to fetch data from remote storage repeatedly, thus improving performance.

Stages

Locations in Snowflake where data files can be stored before being loaded into tables. Stages can be internal (within Snowflake) or external (e.g., AWS S3).

An internal stage can be created using *CREATE STAGE my_stage;*.
Data files can be uploaded to this stage before being loaded into a table.

COPY INTO

Command

Used to load data from a stage into a Snowflake table. The command specifies the target table and the source file(s) along with optional transformations.

Loading data from a stage into a table:
COPY INTO my_table
FROM @my_stage/file.csv
FILE_FORMAT =(FORMAT_NAME = 'my_csv_format');.

Time Travel

Allows users to query, clone, or restore data to a previous state within a defined retention period. This feature aids in data recovery and auditing.

Querying a table as it was at a specific point in time:

```
SELECT *FROM my_table
AT (TIMESTAMP =>'2022-06-01T00:00:00');.
```

Zero-Copy Cloning

Enables creating a clone of a database, schema, or table without copying the data. Changes to the clone do not affect the original, and vice versa.

Creating a clone of a table:
CREATE CLONE my_table_clone OF my_table;.

This allows working with a snapshot of the data without additional storage costs.

Secure Data Sharing

Allows sharing of data across different Snowflake accounts without moving or copying the data. Consumers can query shared data in real-time.

Sharing data with another Snowflake account:

CREATE SHARE my_share;

GRANT SELECT ON TABLE my_table TO SHARE my_share;.

The recipient can access the shared data directly.

Snowsight

Snowflake's new web user interface that enhances the user experience with features like integrated dashboards, interactive visualizations, and an improved SQL editor.

Users can create and manage interactive dashboards within Snowsight, allowing them to visualize data trends and share insights with their team. For example, a sales team can use Snowsight to build a dashboard that tracks monthly sales performance across different regions.

Snowflake

Community

A vibrant network of users, experts, and partners who share knowledge, best practices, and support each other in using Snowflake. It includes user groups, forums, and special interest groups.

Joining the Snowflake Community allows users to participate in discussions, attend meetups, and access valuable resources. For instance, a data analyst can join a virtual special interest group focused on data warehousing to learn from others' experiences and share their own insights.

Data Marketplace

The Snowflake Data Marketplace is a platform where users can discover, access, and share live data sets from various providers. It facilitates data collaboration and allows users to enrich their own data with external data sources.

A marketing team can access demographic data from a third-party provider through the Data Marketplace to enhance their customer analysis. They can integrate this data with their internal sales data to gain deeper insights into customer behavior and preferences.

Multi-Cluster Warehouses

Multi-Cluster Warehouses allow Snowflake to automatically manage the number of compute clusters needed to handle varying workloads. This ensures optimal performance and resource utilization without manual intervention.

A retail company can set up a multi-cluster warehouse to handle the high concurrency of queries during Black Friday sales. Snowflake automatically adds clusters to manage the increased load and removes them when the load decreases, ensuring efficient use of resources and cost management.

Materialized Views

Materialized views store the result set of a query physically and automatically update when the underlying data changes. They improve query performance by providing pre-computed results.

Creating a materialized view:
CREATE MATERIALIZED VIEW mv_sales
*AS SELECT *FROM sales*
WHERE year =2022;.
Queries on this view are faster since the results are pre-computed.

Task

Tasks are used to automate the execution of SQL statements, including procedural logic, at specified intervals or upon completion of other tasks.

Creating a task to run a query every hour:

```
CREATE TASK hourly_task
WAREHOUSE ='my_warehouse'
SCHEDULE ='1HOUR'
AS I
NSERT INTO daily_sales
SELECT *FROM sales
WHERE sales_date =CURRENT_DATE;.
```

Stream

Streams track changes to a table (inserts, updates, deletes) and provide a change data capture (CDC) mechanism for efficient data processing.

Creating a stream:
CREATE STREAM sales_stream ON TABLE sales;.

The stream captures changes to the sales table, which can be processed later.

Pipe

Pipes automate data loading by continuously ingesting data from external stages (e.g., AWS S3, Azure Blob Storage) into Snowflake tables.

Creating a pipe to load data from an S3 bucket:

```
CREATE PIPE my_pipe
AS COPY INTO my_table
FROM @my_stage/file.csv
FILE_FORMAT =(FORMAT_NAME =
'my_csv_format');.
```

Warehouse

Monitoring

Snowflake provides tools to monitor the performance and usage of virtual warehouses, helping users optimize resource allocation and manage costs.

Using the WAREHOUSE_METERING_HISTORY view to monitor warehouse usage and costs:

*SELECT *FROM*
WAREHOUSE_METERING_HISTORY
WHERE WAREHOUSE_NAME =
'my_warehouse';.

Role-Based Access Control (RBAC)

A security model that restricts access to data and resources based on the roles assigned to users. Snowflake allows fine-grained control over access permissions.

Creating a role and granting privileges:
CREATE ROLE analyst_role;
GRANT SELECT ON DATABASE sales_data TO ROLE analyst_role;.

Assigning the role to a user:
GRANT ROLE analyst_role TO USER john_doe;.

Dynamic Data Masking

Dynamic Data Masking allows Snowflake to hide sensitive data in query results based on the role of the user accessing the data. This enhances data security and privacy.

Masking sensitive data:

```
CREATE MASKING POLICY ssn_mask
AS (val STRING)
RETURNS STRING ->CASE
WHEN CURRENT_ROLE() IN ('analyst_role')
THEN 'XXX-XX-XXXX'
ELSE val END; Applying the policy:
ALTER TABLE customers
MODIFY COLUMN ssn
SET MASKING POLICY ssn_mask;.
```

External Tables

External tables allow Snowflake to query data stored in external locations (e.g., AWS S3, Azure Blob Storage) without loading it into Snowflake.

Creating an external table:
CREATE EXTERNAL TABLE my_ext_table WITH LOCATION ='@my_external_stage' FILE_FORMAT =(FORMAT_NAME = 'my_csv_format');.
This table allows querying data directly from the external stage.

Data Replication

Snowflake's data replication feature allows for the replication of databases across different regions and cloud providers to enhance data availability and disaster recovery.

Setting up data replication:
CREATE REPLICATION GROUP my_replication AS REPLICATION TO REGION 'aws_us_west_2';.
This replicates the database to a different AWS region.

Failover and Failback

Snowflake provides failover and failback capabilities to ensure high availability and disaster recovery. Failover allows switching to a replica in case of a failure, and failback switches back once the original is restored.

Configuring failover for a database: *ALTER DATABASE my_database SET FAILOVER GROUP =my_failover_group;*. This ensures that the database can switch to a replica in case of a failure.

Search Optimization

Service

A Snowflake feature that improves the performance of searches on large tables by creating and maintaining search optimization structures.

Enabling search optimization for a table: *ALTER TABLE my_table SET SEARCH OPTIMIZATION =TRUE;*.

This improves the performance of search queries on the table.

Snowflake Data Exchange

A platform that allows Snowflake users to share and access live data securely. It facilitates data collaboration and monetization by providing a marketplace for data providers and consumers.

Publishing data to the Data Exchange: *CREATE EXCHANGE my_exchange; GRANT SELECT ON TABLE my_table TO EXCHANGE my_exchange;*.
Other users can subscribe to and query the shared data.

Data Masking

Data masking provides a way to protect sensitive data by masking it in query results, based on user roles. This ensures that sensitive information is not exposed to unauthorized users.

Creating a data masking policy:

```
CREATE MASKING POLICY email_mask
AS (val STRING)
RETURNS STRING ->CASE
WHEN CURRENT_ROLE() IN ('analyst_role')
THEN '*********@domain.com' ELSE val END;
```

Applying the policy to a column:

```
ALTER TABLE users
MODIFY COLUMN email SET MASKING POLICY
email_mask;.
```

Snowpipe

Snowpipe is Snowflake's continuous data ingestion service, which allows for the automated loading of data from external stages into Snowflake tables.

Creating a Snowpipe to load data:
CREATE PIPE my_pipe
AS COPY INTO my_table
FROM @my_stage
FILE_FORMAT =(FORMAT_NAME = 'my_csv_format');.
Snowpipe will automatically load new data files as they arrive in the stage.

External Functions

External functions allow Snowflake to call external services and integrate with external systems directly from SQL queries. This enables advanced data processing and integration capabilities.

Creating an external function:
CREATE EXTERNAL FUNCTION my_ext_function()
RETURNS STRING API_INTEGRATION = my_api_integration;.
This function can call an external API and return the result to Snowflake.

Streams and Tasks

Streams track changes to tables, and tasks automate the execution of SQL based on schedules or events. Together, they enable efficient change data capture and automation.

Creating a stream and task:

```
CREATE STREAM my_stream
ON TABLE my_table;

CREATE TASK my_task WAREHOUSE =
'my_warehouse' SCHEDULE ='1 HOUR'
AS
INSERT INTO my_target_table
SELECT *FROM my_stream;.
```

Snowflake

Organizations

Snowflake Organizations provide a way to manage multiple Snowflake accounts within an organization. This enables better resource allocation, cost management, and governance.

Creating an organization:
CREATE ORGANIZATION my_org;
and adding accounts to it. This allows central management of multiple Snowflake accounts.

Data Governance

Snowflake offers features for data governance, including access controls, data masking, and audit logging, to ensure data security, privacy, and compliance.

Implementing data governance:
CREATE ROW ACCESS POLICY my_policy AS (val STRING)
RETURNS BOOLEAN ->CURRENT_ROLE()
IN ('data_governance_role');
and applying it to a table.

Account Usage

Snowflake provides account usage views to track and analyze resource usage, query performance, and cost management. These views help in monitoring and optimizing Snowflake usage.

Querying account usage:
*SELECT *FROM ACCOUNT_USAGE.QUERY_HISTORY WHERE QUERY_TEXT ILIKE '%SELECT%';*
This retrieves the history of SELECT queries executed in the account.

Resource Monitors

Resource monitors allow administrators to manage and control compute resource usage by setting thresholds and triggering actions when limits are reached.

Creating a resource monitor:
CREATE RESOURCE MONITOR my_monitor WITH CREDIT_QUOTA =1000;
and assigning it to a warehouse. This monitor will track the compute credits used by the warehouse and take action if the quota is exceeded.

Query Optimization

Snowflake provides various tools and techniques to optimize query performance, including using the Query Profiler, optimizing table structures, and leveraging caching.

Using the Query Profiler:

```
SELECT *FROM
TABLE(QUERY_HISTORY_BY_SESSION(SESSI
ON_ID =>'my_session'));
```

This helps identify and optimize slow- running queries.

Data Sharing

Snowflake allows secure sharing of data between different accounts without data movement. Shared data can be accessed in real-time, ensuring consistency and reducing latency.

Creating a share: *CREATE SHARE my_share;* and adding tables to it. Other Snowflake accounts can access the shared data directly.

Cloning

Cloning in Snowflake creates a copy of a database, schema, or table without duplicating the data. This is useful for creating test environments and for backup purposes.

Cloning a table:
CREATE CLONE my_table_clone OF my_table;
This allows working with a snapshot of the data without additional storage costs.

Data Load and Unload

Snowflake provides various methods for loading and unloading data, including bulk loading with the COPY command, using Snowpipe for continuous loading, and unloading data to external stages.

Loading data:
COPY INTO my_table
FROM @my_stage FILE_FORMAT =
(FORMAT_NAME ='my_csv_format');
and unloading data:
COPY INTO @my_stage FROM my_table;.

Data Encryption

Snowflake encrypts data at rest and in transit to ensure data security. Encryption keys are managed automatically, and users can also provide their own keys for additional security.

Enabling encryption for a table:
ALTER TABLE my_table SET DATA_RETENTION_TIME_IN_DAYS =90;
This ensures that data is encrypted and retained for a specified period.

Data Retention

Snowflake provides data retention policies to manage how long data is kept in the system. This includes Time Travel and Fail-safe periods for data recovery.

Setting data retention:
ALTER TABLE my_table SET DATA_RETENTION_TIME_IN_DAYS =7;
This configures the table to retain historical data for 7 days.

Fail-Safe

Fail-Safe is a Snowflake feature that provides an additional 7-day period for recovering data after the Time Travel retention period has expired. This ensures data recovery in case of failures.

Accessing Fail-Safe data:
*SELECT *FROM my_table BEFORE (END_TIME =>'2022-06-01T00:00:00');*
This retrieves data that is in the Fail-Safe period.

User-Defined Functions (UDFs)

UDFs allow users to define their own functions in SQL or JavaScript, extending Snowflake's built-in functionality with custom logic.

Creating a SQL UDF:
CREATE FUNCTION my_udf(x INT)
RETURNS INT
LANGUAGE SQL
AS
*'RETURN x *2';*
This function multiplies the input by 2.

Stored Procedures

Stored procedures in Snowflake allow for procedural logic and complex operations to be encapsulated in SQL or JavaScript, enabling automation and reusable code.

Creating a stored procedure:

```
CREATE PROCEDURE my_proc()
RETURNS STRING LANGUAGE JAVASCRIPT
AS $$ return 'Hello, World!';
$$;
```

and calling it:

```
CALL my_proc();.
```

Privileges and Grants

Snowflake's security model uses privileges and grants to control access to database objects. Roles are assigned privileges, and users are assigned roles.

Granting privileges:
GRANT SELECT ON TABLE my_table TO ROLE analyst_role;
This allows users with the analyst_role to query the table.

Roles and Role Hierarchies

Roles in Snowflake define a set of privileges and can be assigned to users. Role hierarchies allow roles to inherit privileges from other roles, simplifying access management.

Creating a role hierarchy:
CREATE ROLE senior_analyst;
GRANT ROLE analyst_role TO ROLE senior_analyst;
Users with the senior_analyst role inherit privileges from the analyst_role.

Session Variables

Session variables in Snowflake store values that can be used within a session. They allow for dynamic SQL and reusable code.

Setting and using a session variable:
SET my_var ='Hello, World!';
and *SELECT $my_var;*
This returns the value of the variable.

Parameter Management

Snowflake allows configuration of various parameters at the account, session, and object levels to customize behavior and optimize performance.

Setting a session parameter:
ALTER SESSION SET QUERY_TAG = 'MyQuery';
This tags queries within the session for easier tracking.

Semi-Structured Data

Snowflake supports semi-structured data formats such as JSON, Avro, Parquet, and XML. This allows for flexible data modeling and integration with modern data sources.

Querying JSON data: SELECT json_data:id FROM my_table;. This retrieves the "id" field from JSON data stored in a column.

Data Compression

Snowflake automatically compresses data to reduce storage costs and improve query performance. Different compression algorithms are used based on the data type.

Snowflake's automatic compression means users don't need to manually configure compression settings, as the platform optimizes storage efficiency.

Cost Management

Snowflake provides tools and practices to manage and optimize costs, including resource monitors, usage views, and best practices for query optimization.

Using resource monitors to control costs:
CREATE RESOURCE MONITOR my_monitor WITH CREDIT_QUOTA =1000;
and setting up alerts for budget thresholds.

Query History

Snowflake tracks query history, allowing users to review and analyze past queries for performance optimization and troubleshooting.

Accessing query history:
*SELECT *FROM QUERY_HISTORY WHERE QUERY_TEXT ILIKE '%SELECT%';* This retrieves a history of SELECT queries executed in the account.

Metadata Management

Snowflake manages metadata for all database objects, providing detailed information about tables, columns, and other objects. This metadata is used for query optimization and data governance.

Querying metadata:
*SELECT *FROM*
INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA ='PUBLIC';
This retrieves information about all tables in the PUBLIC schema.

Data Import/Export

Snowflake supports various methods for importing and exporting data, including bulk loading with the COPY command and unloading to external stages.

Importing data:

```
COPY INTO my_table
FROM @my_stage FILE_FORMAT =
(FORMAT_NAME ='my_csv_format');
```

and exporting data:

```
COPY INTO @my_stage FROM my_table;
```

Data Quality

Snowflake provides features to ensure data quality, including constraints, data validation, and profiling.

Implementing data quality checks:

```
CREATE TABLE my_table (
id INT PRIMARY KEY,
name STRING NOT NULL);
```

This ensures that the "id" column is unique and the "name" column is not null.

Data Lineage

Data lineage tracks the flow of data through Snowflake, from ingestion to transformation to analysis, providing visibility into data dependencies and transformations.

Using views and tasks to track data lineage:

```
CREATE VIEW my_view
AS
SELECT *FROM my_table;
```

and

```
CREATE TASK my_task
AS
INSERT INTO my_table
SELECT *FROM my_view;
```

Business Continuity

Snowflake's features for business continuity include data replication, failover, and fail-safe, ensuring that data is always available and recoverable in case of disasters.

Setting up a failover group:
CREATE FAILOVER GROUP my_group
AS FAILOVER TO REGION 'aws_us_west_2';
This ensures that the database can switch to a replica in case of a failure.

Governance and Compliance

Snowflake provides tools for data governance and compliance, including access controls, data masking, and audit logging, to ensure data security and regulatory compliance.

Implementing compliance policies:

```
CREATE ROW ACCESS POLICY
compliance_policy AS (val STRING)
RETURNS BOOLEAN ->CURRENT_ROLE()
IN ('compliance_role');
```

and applying it to a table.

Advanced Analytics

Snowflake supports advanced analytics capabilities, including machine learning integration, geospatial data processing, and complex data transformations.

Integrating with machine learning models:
CREATE FUNCTION predict_sales(x FLOAT)
RETURNS FLOAT
LANGUAGE PYTHON RUNTIME ='3.8'
HANDLER ='my_model.predict';
This function calls a Python model for sales prediction.

Data Monetization

Snowflake's data marketplace and secure data sharing enable organizations to monetize their data assets by sharing or selling data to other Snowflake users.

Publishing data for monetization:
CREATE EXCHANGE my_exchange;
and adding data to it for other users to access and purchase.

Geospatial Data

Snowflake supports geospatial data types and functions, allowing users to store, query, and analyze spatial data such as points, polygons, and geometries.

Querying geospatial data:
SELECT ST_DISTANCE(point1, point2)
FROM my_table;
This calculates the distance between two points stored in a table.

IoT Data Processing

Snowflake's scalable architecture and support for semi-structured data make it well-suited for processing and analyzing IoT (Internet of Things) data.

Loading IoT data:
COPY INTO my_table
FROM @iot_stage
FILE_FORMAT =(FORMAT_NAME = 'json_format');
This ingests JSON data from IoT devices.

Real-Time Analytics

Snowflake supports real-time analytics by allowing continuous data ingestion and immediate querying of fresh data.

Using Snowpipe for real-time data ingestion:

```
CREATE PIPE my_pipe
AS COPY INTO my_table
FROM @my_stage
FILE_FORMAT =(FORMAT_NAME =
'my_csv_format');
```

Data Federation

Snowflake's external tables and data sharing features enable data federation, allowing users to query and combine data from multiple sources without moving the data.

Creating an external table to federate data: *CREATE EXTERNAL TABLE my_ext_table WITH LOCATION ='@my_external_stage' FILE_FORMAT = (FORMAT_NAME = 'my_csv_format');*

This table allows querying data directly from the external stage.

Security

Integrations

Snowflake integrates with security tools and frameworks, including single sign-on (SSO), multi-factor authentication (MFA), and encryption key management, to enhance data security.

Configuring SSO:
ALTER ACCOUNT SET SSO_LOGIN_PAGE = 'https://mycompany.com/sso';.
This enables single sign-on for Snowflake users.

Continuous Data Protection

Snowflake's continuous data protection features include Time Travel, Fail-safe, and data replication, ensuring data integrity and availability at all times.

Setting up data replication:
CREATE REPLICATION GROUP my_replication AS REPLICATION TO REGION 'aws_us_west_2';.
This replicates the database to a different AWS region.

Custom Data Types

Snowflake allows users to define custom data types and enforce data integrity through constraints and validation rules.

Creating a custom data type:
CREATE DOMAIN email_type
AS STRING CHECK (VALUE LIKE '%@%.%');
This enforces email format validation.

Hybrid Tables

Hybrid tables in Snowflake combine the benefits of transactional and analytical processing, allowing for efficient real-time data analysis.

Creating a hybrid table:
CREATE HYBRID TABLE my_table
(id INT, data STRING);

This table supports both transactional and analytical workloads.

Data Archiving

Snowflake's data retention and archiving features help manage long-term storage of historical data, ensuring that it is available for compliance and analysis.

Setting data retention:
ALTER TABLE my_table
SET DATA_RETENTION_TIME_IN_DAYS =365;
This configures the table to retain historical data for one year.

Data Classification

Data classification in Snowflake helps categorize data based on sensitivity and importance, enabling better data governance and security.

Classifying data:
ALTER TABLE my_table SET TAG classification ='sensitive';
This tags the table as containing sensitive data.

Data Masking

Policies

Data masking policies in Snowflake provide dynamic masking of sensitive data based on user roles, ensuring that only authorized users can see the actual data.

Creating a data masking policy:

```
CREATE MASKING POLICY ssn_mask
AS (val STRING)
RETURNS STRING ->CASE
WHEN CURRENT_ROLE() IN ('analyst_role')
THEN 'XXX-XX-XXXX' ELSE val END;
```

Applying the policy:

```
ALTER TABLE customers MODIFY COLUMN ssn
SET MASKING POLICY ssn_mask;
```

Row Access Policies

Row access policies allow Snowflake to restrict access to specific rows in a table based on user roles and other criteria, enhancing data security and compliance.

Creating a row access policy:

CREATE ROW ACCESS POLICY row_policy
AS (val STRING)
RETURNS BOOLEAN ->CURRENT_ROLE() IN ('analyst_role');

Applying the policy:

ALTER TABLE my_table MODIFY ROW ACCESS POLICY row_policy;.

Cross-Cloud Replication

Snowflake supports cross-cloud replication, allowing data to be replicated across different cloud providers (e.g., AWS, Azure, Google Cloud) for high availability and disaster recovery.

Setting up cross-cloud replication:
CREATE REPLICATION GROUP my_replication
AS REPLICATION TO REGION 'azure_eastus';
This replicates the database to an Azure region.

Event-Driven Data Processing

Snowflake's tasks and streams enable event-driven data processing, allowing actions to be triggered based on changes in data or scheduled intervals.

Creating an event-driven task:

```
CREATE TASK my_task
WAREHOUSE ='my_warehouse'
AFTER INSERT ON my_table
AS I
NSERT INTO audit_table
SELECT *FROM my_table;
```

Data Encryption Key Management

Snowflake allows users to manage their own encryption keys for added security, providing control over data encryption and compliance with regulatory requirements.

Setting a customer-managed key:
ALTER DATABASE my_database SET ENCRYPTION ='my_custom_key';

This uses a user-provided key for data encryption.

Geospatial Functions

Snowflake provides geospatial functions to perform spatial analysis and operations on geographic data, such as distance calculations and spatial joins.

Using a geospatial function:
SELECT ST_DISTANCE(point1, point2)
FROM my_table;
This calculates the distance between two geographic points stored in a table.

Graph Analytics

Snowflake supports graph analytics, enabling users to model and analyze relationships between data points using graph structures and algorithms.

Performing graph analytics:
CREATE TABLE graph_edges
(src INT, dst INT);
and running graph queries to analyze relationships.

Data Versioning

Snowflake's Time Travel and Zero-Copy Cloning features enable data versioning, allowing users to create, manage, and query different versions of data for analysis and auditing.

Creating a version of a table:
CREATE CLONE my_table_clone
OF my_table;

This clone represents a version of the original table that can be queried and analyzed separately.

API Integration

Snowflake supports integration with external APIs, allowing users to call external services and incorporate real-time data into Snowflake queries and workflows.

Creating an external function to call an API:
CREATE EXTERNAL FUNCTION my_ext_function() RETURNS STRING API_INTEGRATION =my_api_integration;
This function can call an external API and return the result to Snowflake.

THANK YOU