Snowflake Architecture

- Cloud Native, Multicluster, Shared Data
- DataWarehouse Platform
- Supported Cloud platforms - AWS, Azure, GCP Regions
- SAAS
- OLAP, Recommends ELT
- ACID compliant
- Versions available: Standard, Enterprise, Business Critical, VPS
- **Snowflake layers**
    - Storage layer - Actual database
    - Compute layer - Query processing
    - Cloud Services layer - Infrastructure, metadata, query compilation and optimization, security
    - Cloud Agnostic Layer
- **Data storage** – Compressed (algorithm can be different for different columns, based on data analysis), Columnar format
- **Authentication -** MFA and Federated auth is available in all editions
- **Storage layer** - microPartitions (50 - 500MB uncompressed data), 16 MB in compressed format.
- **E2E encryption -** All data, at rest or in transit is encrypted by default
    - At rest - AES-256
    - In transit - TLS1.2, HTTPS
- Database and share replication available in all editions
- Failover/Failback mechanism is available in Business Critical and VPS
- Can work with on-premises tools via connectors like jdbc/odbc connectors.
- Continuous Data Protection (CDP):
    - Time Travel, fail safe
    - Network policies
    - MFA , SSO
    - Access Policies - DAC, RBAC, roles, privileges

Datatypes

- **Standard Datatypes**
  - int
  - Binary / VarBinary - 8MB - Each byte represents 2 hex characters
  - Float, double, DoublePrecision, Real
  - String, Varchar - 16MB
  - Boolean -
    - True - 'true', 't', '1', 'on', 'y', 'yes'
    - False - 'false', 'f', '0', 'off', 'n', 'no'
  - Date Time / TimeStamp
  - Datetime / TimeStamp_NTZ
  - TimeStamp_LTZ
  - TimeStamp_TZ
  - Null
  - GeoSpatial
    - Geography - geospatial data represented as degrees of latitude and longitude
    - Geometry - geospatial data in a planar coordinate system
- **Semi-Structured Datatypes**
  - Variant
    - Allows storage of semi-structured data in a single column
    - Also called as variant / universal datatype
    - Can hold any type of data object / array
    - Stored internally as compressed, columnar, binary representation
    - Max size 16MB compressed

    Datatypes supported :

    - Object
    - Array
    - JSON
    - XML
    - Other datatypes
    - **CSV, TSV**
    - **JSON** - SQL query supported, Notation **- ':'** - src:studId / **'[]'** - src[studId]
    - **AVRO** – Binary, SQL query supported
    - **ORC** – Binary, SQL query supported

- **Parquet** – Binary, SQL query supported
  - Parquet does not truncate floating point numbers, CSV and JSON truncates
  - Parquet file - max size allowed - 5 GB
- **XML** - SQL query NOT supported
  - Use parse_xml function to convert xml to string and then string manipulation functions
  - Use javascript to process xml
  - Use third party libraries

Variant datatype Functions:

- Parse_json - converts a string to a JSON compatible variant
- To_json - converts a variant to a string
- Check_json
- object_construct
- Typeof
- Json_extract_path_text
- Strip_null_value
- strip *outer*_array - turns a large JSON into small JSONs.
- Flatten
  - Explodes complex data into multiple rows
  - For better pruning and less storage consumption
  - If JSON file contains dates, time stamps, arrays, numbers in strings
  - Output of flatten - Seq, Path, Value, Key, Index
  - Options - Lateral and Table

Null handling in Variant columns:

- SQL NULL: NULL
- JSON null: stored as strings

Virtual Warehouses

- First class object
- Compute layer, query processing layer
- MPP - Clusters (Servers / nodes)
- Sizes – x-Small – 6XL
- VW size applies to all clusters in a warehouse.
- Standard warehouse from webUI - X-Large
- Standard warehouse from snowsql - X-Small
- X-small – 1 server, 1 credits / hour
- Cache / Local disk cache - 14 days
- No of queries a warehouse can execute depends on
    - Size of data required for each query
    - Complexity of the query
    - MAX_CONCURRENCY_LEVEL parameter also has an impact

Resizing of warehouses

- **Scale up**
    - Increasing capacity of a single server (XL to 2 XL)
    - To accommodate complex / large queries, not automatic
    - When performance of slow running and very large complex queries needs improvement , or if bytes are spilled to local storage
    - Impacts only subsequent queries or queued queries, not currently running ones.
    - Decreasing size of warehouse would delete some local cache as some compute resources are removed in the process.
- **Scale out**
    - Adding more servers to a cluster keeping capacity same
    - To accommodate concurrent / many queries or users. Reduce queues

Factors to consider while resizing a warehouse

- no of queries
- no of tables
- data size and composition

Elastic DataWarehousing - Multi Cluster warehouses

- Standard and snowpark optimized clusters
    - Single cluster
    - Multi Cluster - Enterprise edition and above

    - Max number of cluster servers - 10
    - Scaling modes - Maximised mode, Auto Scale mode
    - Scaling Policy (AutoScale - ON):
        - Standard:
            - Prefers less load on clusters than queuing
            - Next cluster is started when a query is queued
            - Cluster is shut down after 2 -3 checks of validating if load on least loaded cluster can be redistributed to other clusters without starting the cluster again (1 min diff)
            - Warehouse starts 20 secs after the first cluster starts
        - Economy:
            - Prefers queuing over less loading of clusters
            - Starts new cluster only if there is enough load to keep it busy for 6 mins
            - Shuts down clusters after 5-6 checks
            - Warehouse starts only if there is enough load to keep the cluster busy for 6 mins
            - Multi clustering used when concurrency is to be improved (ability to process more queries)
- Warehouse starts when 100% servers start. In case some of the servers in the warehouse fail to start.
- Atleast 50% of the servers of the warehouse have to be provisioned before a warehouse can start running queries.
- WAIT_FOR_COMPLETION parameter of Alter warehouse- this command returns only after complete provisioning of the warehouse is done after resizing. - indicates that the warehouse is ready to start its execution.
- **Total Query load:** Execution time for running queries + time for which queries were queued / time duration of interval
- Query history is maintained for 14 days.

Virtual Warehouse Functions / parameters:

- MAX_CONCURRENCY_LEVEL - no of queries that can run concurrently
- STATEMENT_TIMEOUT_IN_SECONDS - max time that a query can run
- STATEMENT_QUEUED_TIMEOUT_IN_SECONDS - max time that a query can stay in a queue run
- CURRENT_WAREHOUSE - warehouse in current session

Privileges:

- Operate - Starting stopping warehouses, Aborting executing queries
- Monitor - monitor performance and extract usage metrics
- Modify - to resize a warehouse
- Usage
- Applybudget
- Ownership

Billing:

- Billing happens per compute resource - size of cluster
- Size of warehouse
- Region where warehouse is located

TimeTravel

Time travel is a read only point in time snapshot of data.

- Snowflake maintains data pertaining to changed records only, not a full copy.
- Full copy of table data is taken when table is dropped / truncated.
-
- Cannot be disabled for accounts but : DATA_RETENTION_TIME_IN_DAYS can be set to 0.
- DATA_RETENTION_TIME_IN_DAYS - default 1 for all editions
- Supported on databases, schemas, tables
- If you change the data retention period for a table, the new retention period impacts all data that is active, as well as any data currently in Time Travel
- Data cannot be retrieved back once it moves into fail safe even if time travel period is changed.
- Can undrop account.
- Standard Edition: Retention period : Default 1 day (24 hours)
    - Automatically enabled for all Snowflake accounts
    - Permanent: 0 -1 days
    - Transient: 0 -1 days
    - Temporary: 0 -1 days
- Enterprise Edition (and higher): Retention period : Default 1 day (24 hours)
    - Permanent tables : 0 up to 90 days.
    - Transient and temporary tables : Max retention period : 1 day
- AT | BEFORE clause:

    - BEFORE : brings data immediately preceding the before statement not including the statement.
    - Can be specified in SELECT / CLONE / CREATE statements
    - TIMESTAMP
    - STATEMENT (query id)
    - OFFSET (time difference in seconds, starting current time)
- Parameters
    - DATA_RETENTION_TIME_IN_DAYS
    - MIN_DATA_RETENTION_TIME_IN_DAYS - set at Account level : effective retention time calculation is : MAX(DATA_RETENTION_TIME_IN_DAYS, MIN_DATA_RETENTION_TIME_IN_DAYS).
- Not supported on Materialized views.
- Not supported for External tables.

FailSafe

- 7 days non configurable
- Can be retrieved by snowflake support only
- Permanent objects 7 days.
- Transient and temporary - no fail safe period, 0 days
- Additional cost is incurred
- Data cannot be retrieved back once it moves into fail safe even if time travel period is changed.

Databases in Snowflake

- Row size limit - 16 MB
- INSERT and COPY statements are not blocking, but update, merge and delete are
- Permanent tables
- Temporary tables
    - Deleted when session is deleted, no fail safe, not visible to other users, can have same name as permanent tables.
    - In conflicts of same names for permanent and temp tables, Temporary tables take precedence when queried in a session
    - Time travel – max 1 day
    - no fail safe
    - Not replicated
    - Can be cloned to another temporary table only, but only within same session
    - Clone will also get deleted when session ends
    - Example: Temporary storage for ETL queries.

- Transient tables
    - Time travel – max 1 day
    - no fail safe
    - When to use : As part of a data processing pipeline, you are required to store data in an interim table. The subsequent processes then use the table in the pipeline. The data is deleted and reloaded every time the pipeline is executed. You are required to minimize data storage costs.
- External tables
    - Supported formats: AVRO, ORC, Parquet, CSV, JSON
    - do not support XML
    - Read only
    - Time travel is not available
    - Created only on external stages.
    - Table is outside of snowflake
    - Can create materialized views to improve performance
    - Refresh of metadata contributes to event notification costs. Hence contribute to costs
    - Columns:
        - Value (Variant),
        - metadata$Filename (Staged filename and path),
        - metadata$File_Row_Number (row number for each record stored in stage file)
- Directory tables - Catalog of staged files
- Hybrid tables
- Iceberg tables
- Dynamic tables
- Event Tables

Database objects:

- Schemas: Database can contain any number of schemas
- Tables
- Views
- Procedures
- Tasks
- Streams
- Sequences
- File Formats
- Stages

- UDFs
- Future tables

Micro Partitions

- Micro partitions are immutable, contiguous units of data
- Data is stored in compressed and encrypted forms
- Each micro partition stores 50 - 500 MB of uncompressed data
- Storage cost is based on compressed file size
- Cost is calculated on monthly basis.
- Flat rate of /TB of data
- Snowflake stores metadata about all rows stored in a micro-partition, including:
    - The range of values for each of the columns in the micro-partition.
    - The number of distinct values.
    - Additional properties used for both optimization and efficient query processing.
    - Clustered values overlap
    - Size

Views

A view is a wrapper/query around a table / tables to extract data and represent it in a simplistic format with enhanced data security.

- Views can be defined on tables, Streams
- To change view definition, the view needs to be recreated. It cannot be altered.
- Standard View
    - Do not cache data
    - No storage costs
    - Can be created on permanent tables, external tables, temporary tables, etc
    - Time travel
- Secure View
    - Only Secure views can be shared
    - They can be built on multiple tables form various databases as long as databases are in the same account
    - Secure views are accessible to only those users with ownership privileges
    - Internal optimizations are not available
    - No details of query plan are available in query profilematerialioptimi
- Materialized View
    - Require enterprise edition.

    - Needs a running warehouse
    - Can be created on
        - Permanent tables
        - External tables
    - Data is cached. Hence attributes to storage costs.
    - If underlying data in the table changes, materialized view would use the unchanged data from its cache and query table only for changed data.
    - Can query only a single table
    - Cannot include group_by, joins, limit, having clauses
    - Does not support many aggregate functions.
    - Supports, min, max, avg, sum, etc, only a few
    - Standard DML insert, update, delete not allowed on these views
    - Clustering can be applied
    - Background process keeps materialized view updated with changed data.
    - Removing columns from base table will suspend all materialized views. These materialized views have to be recreated.
    - Deleting base table will suspend materialized view
    - Can apply masking policies
    - Cannot create clones directly, but if DB with MV is cloned, it is allowed.
    - Create view privileges - “CREATE MATERIALIzED VIEW’ on schema to ROLE
- Recursive view
    - Only standard / non-materialized views can be defined as recursive
    - It is a join which refers the same view.

Stored Procedures

- Procedure can be defined as a CALLER or OWNER (default - OWNER)
- Nested stored Procedures are possible. Each nested procedure is an independent transaction. Nested transactions are not supported in snowflake.
- Stored procedure can return only 1 value or 1 table
- It is possible to not return any value i.e. a stored procedure without a return statement.
- Can return a single value or a tabular data
- Needs a running warehouse

Characteristics	UDF	UDTF	External Function	Stored Proc
Returns value	Single value	Multiple values	Single value	Single or tabular
Can be called from SQL	Y	Y	Y	Y

Functions

Types:

- UDF
  - SQL, Javascript, Java, Python, Scala
- Secure UDF - bypasses all optimisations. Hence less performant.
- UDTF – returns more than one value
- External Functions
  - Data base object
  - Code resides outside of snowflake.(AWS lambda function, Azure Function, Node.js script running on ec2)
  - Are executed outside of snowflake.
  - Schema object, created within snowflake to call this service.
  - Supports only scalar functions (returning one value)

Tasks

Tasks can be imagined as schedulers which can perform actions like copying data from a stage to a table periodically / as per the schedule specified in create task command.

- Tasks are only schedule based not event based.
- Cannot be invoked independently, can be called from a statement / stored proc of a task.
- To start a task, after it is created, we need to explicitly ALTER TASK and set it to RESUME.
- Tasks can also be SUSPEND ed.
- Tree of tasks - max 1000 levels

- Child Tasks - 100 max
- Predecessor tasks 100 max
- Task can have multiple predecessor tasks.
- Snowflake account is limited to 10000 resumed tasks.
- To define a child task - AFTER keyword in create task command
- Can execute a single SQL statement only
- Can call a stored procedure
- Can execute a function if its inside a Stored proc in javascript but not java and python
- Only Root Tasks can define a schedule, not child tasks.
- Only one instance of a DAG (task) can execute at a time with a schedule in an account.
- Task DAG Cannot span multiple schemas
- All tasks in a DAG need to be a part of same database and schema.
- All tasks in a DAG need to have a single owner (role).
- If a role is dropped, it completes it exec under dropped role.
- Tasks can be cloned
- Require compute resources to execute SQL statements (Can be snowflake managed / VWH)
- Timeout for a task - 60 mins

Task Functions:

- SYSTEM$CURRENT_USER_TASK_NAME : Returns name of current executing task.
- TASK_HISTORY : Table function in Information_Schema. Returns history of last 7 days
- TASK_DEPENDENTS : List of Child Tasks
- SYSTEM$SET_RETURN_VALUE - to return a value from a task

DDL commands:

- CREATE TASK
- ALTER TASK
- DROP TASK
- DESCRIBE TASK
- SHOW TASKS
- To suspend / resume task : ALTER TASK [IF EXISTS] <name> RESUME | SUSPEND
- To execute a task : EXECUTE TASK <name>

Privileges:

- Account - Execute Managed Task (for serverless)
- Database - USAGE
- Schema - USAGE, CREATE TASK
- Warehouse - USAGE
- To Create a task: Need Account admin or custom role with CREATE TASK privileges
- To Alter a task: Global privilege EXECUTE TASK and OPERATE privilege on the task
- To view a task: Account Admin, task owner , Global Monitor Execution privilege
- To view a task history: Account Admin, task owner , Global Monitor Execution privilege

Streams

Streams can be imagined as audit logs for DML changes that happen on standard tables, directory tables, external tables and views (not materialized views).

- One of the typical usage of stream object is the CDC (Change Data Capture)
- They track actions on rows and not columns.
- Streams do not contain data.
- They store offset for the source table.
- When queried, streams return Historical data from source object
- Multiple streams can be created on a single table. Number is not limited.
- Streams can be used with directory tables. Stream is created on the stage object
- Streams consume storage resources though miniscule.
- Streams have repeatable read isolation - will return same dataset for different queries within a transaction.
- Streams have an exactly once semantics.- stream data can be read only once (reading stream data is called consuming a stream). After this read, stream offset is set to a position right after the record that is read. This becomes the new transactional offset.
- When a stream is created, 3 hidden columns are added to a table.
    - METADATA$ACTION - values : Insert , Delete
    - METADATA$ISUPDATE
    - METADATA$ROWID
    - Streams on Directory tables - METADATA$ROWID - is blank
- Types of Streams:

    - Standard - returns net effect of transactions on rows on standard tables, directory tables and views insert, update, delete and truncate.
    - Append_only - track only inserts on standard tables, directory tables and views
    - Insert_only - track only inserts on external tables
- Alternative to streams is 'changes'. To enable it use:
    - Alter table <tablename> set change_tracking = true
    - E.g. : Select * from table tablename changes(information => default) at(timestamp => <Timestamp_value>);
- Stream becomes stale when its offset is greater than the parameter DATA_RETENTION_TIME_IN_DAYS.
- When stream becomes stale, all records in the stream including unconsumed ones are not accessible.
- To prevent stream from going stale, set MAX_DATA_EXTENSION_TIME_IN_DAYS parameter
- MAX_DATA_EXTENSION_TIME_IN_DAYS - Object parameter, 0-90 days, default 14 days
- Streams can be cloned.
    - Cloning a table does not give access to stream data before cloning.
    - Grants have to be copied explicitly when streams are cloned.
    - current transactional table offset is inherited from source stream.i.e. - unconsumed records are accessible in new table?
- Stream cannot be shared or replicated.

Stream Functions:

- SYSTEM$STREAM_GET_TABLE_TIMESTAMP : returns current offset of the table.
- SYSTEM$STREAM_HAS_DATA
- MAX_DATA_EXTENSION_TIME_IN_DAYS - 0 - 14 days

DDL commands:

- CREATE STREAM
- ALTER STREAM
- DROP STREAM
- DESCRIBE STREAM (STALE and STALE_AFTER column for staleness)
- SHOW STREAMS (STALE and STALE_AFTER column for staleness)

Privileges:

- Database - Usage

- Schema - USAGE, create stream
- Table - Select
- External Table - Select
- View - Select

Cloning

Cones are a point in time snapshot of data that can be updated.

- Databases, schemas, tables, tasks, streams, stages, file formats, sequences, can be cloned.
- Data updated in source after cloning does not reflect in destination (cloned) table and vice versa.
- Clones can be cloned multiple times.
- Clustering keys are cloned
- Cloning databases clones external stages. Does not clone internal stages.
- PIPES can be cloned.
- Pipes referencing external stage can be cloned.
- views and stored procs inside databases get cloned after Databases are cloned. They refer to original tables that they were created on, even after cloning.
- Clauses supported:
  - AT | BEFORE
  - STATEMENT
  - OFFSET
- Cloning tables - select on source table
- What cannot be cloned:
  - Internal stages
  - Pipes on internal stages
  - Transient tables cannot be cloned to permanent tables.
  - Objects cannot be cloned across accounts, cloning happens within same account only.
  - Grants / privileges cannot be cloned.

Clustering

A table with one or more clustering keys defined is called a clustered table

- Supported on all editions
- Clustering - Data is colocated within micropartition for efficient pruning of data
- Re - Clustering is re-grouping or co-locating similar data across micro partitions so that time is saved while extracting data that is a part of a where clause or a group by or a filter condition.
- This process results in creating new micro partitions.
- Clustering consumes compute resources (snowflake provisioned warehouses)
- Single clustering key can contain one or more table columns
- Columns with any datatype are eligible to be included in clustering key EXCEPT - Geography, variant, object, Array
- Only 1 Clustering key per table.
- Defined on large tables (multi terabytes) that are read frequently. Updated less frequently.
- Clustering keys can be defined for
    - Tables (permanent, temporary, transient)
    - Materialized views
- Clustering Depth: Overlapping micro-partitions.
    - A table with no data / micropartitions has a clustering depth of 0
    - A table with no overlapping partitions has clustering depth 1
    - Smaller the cluster depth better is clustering
- Cardinality: no of distinct values, uniqueness
    - High cardinality - makes query highly selective - more cost to maintain clustering
    - Low cardinality - too few partitions - performance will be degraded as pruning will not happen effectively
- Considerations for clustering key definition:
    - Table contains multiple terabytes of data
    - Should be defined on tables which are accessed frequently and change occasionally.
    - Filters like Where, order_by, group_by clauses
    - Join predicates
    - Maximum of 3 or 4 columns (or expressions) per key. Adding more than 3-4 columns tends to increase costs more than benefits.
    - Columns arranged in key from low to high cardinality
- Automatic clustering (Reclustering) –

    - Serverless - shows up in billing as a separate warehouse named AUTOMATIC_CLUSTERING
    - Can be turned off at account level
    - Is a non blocking operation
    - To track clustering billing use view - automatic_clustering_history view
- System Functions:
    - SYSTEM$CLUSTERING_INFORMATION('table', '(col1,col3)')
    - SYSTEM$CLUSTERING_DEPTH
- Privileges:
    - Usage / ownership on schema and database

Caching in Snowflake

Caching happens at 3 places:

- Cloud services layer:
    - Metadata Cache - No of rows in table, column names, distinct values, min and max values
    - Query Results Cache
        - 24 hrs - 31 days
        - Data stored in this cache for 24 hrs
        - Invalidated if underlying table data changes
        - Default - enabled
        - To turn off - account level parameter - USE_CACHED_RESULT - set to false
        - Columns accessed during query are cached
- Local / SSD storage - Compute layer (VWH)
    - Used to cache data used by SQL queries. Whenever data is needed for a given query it's retrieved from the Remote Disk storage, and cached in SSD and memory.
    - Cache is deleted when VWH is suspended / resized
- Remote Storage - Storage layer (Actual database)

Data Share

Shares can be imagined as objects which contain all information to share data from one account to another

- Available on all editions except VPS
- Shares are account level objects.
- Shares are secure, configurable and controlled completely by Provider account.
- Data sharing can be done by : Listings, Direct Share,Data Exchange
    - Listings : Types : Marketplace, free / standard, personalized, paid, private (not available on data marketplace).
    - Direct share : Full Account, Reader Accounts
    - Full Account shares are consumer accounts for consumers who have snowflake accounts - cost on consumer
    - Reader accounts / Managed Accounts to share data with consumers who don't have snowflake accounts. - cost on provider
        - Account admins can create reader accounts
        - Provider can create max 20 Reader accounts.
        - Reader account, 2000 credits / month
- Data Exchange: private centralised data repository.
- Shares created by Producers and imported by consumers.
- Producers - Outbound shares
- Consumers - Inbound Shares
- Data shares between accounts in same region only and on same data providers.
- Shares are read-only, cannot be changed by consumers.
- Data cannot be edited, Comments cannot be edited.
- All db objects shared via shares are Read-Only. Only Select operation is allowed
- Share can include data from multiple databases
- Shared databases cannot be cloned
- Marketplace listing can be accessed by users with privilege of Create database and Import Share.
- Marketplace data is not available for VPS edition
- All data sharing is via secure data sharing.
- Shared objects if are dropped and recreated, grants have to be given explicitly to share them again.
- Grant ownership - ownership of a share cannot be granted to another role after a share is created. Share needs to be recreated with the new role.
- Secure objects are available to users with ownership privileges only.

- Actual data not shared, metadata store and services layer is used for this purpose.
- Since no data is shared, no storage charges for consumer, only compute charges to query data.
- Warehouse is required to query shared data.
- Can create and consume many shares at a time.
- Only 1 database can be created per share.
- Can share to other accounts via views
- Shares cannot be reshared.
- Data Share is integrated with RBAC
- Objects that can be shared via data sharing:
    - Databases
    - Tables
    - Dynamic Tables
    - Iceberg Tables
    - External tables
    - Secure UDFs
    - Secure Views
    - Secure Materialized Views

Privileges:

- AccountAdmin
- Roles with Create_Shares privileges
- Reference_usage privilege on databases to be included in secure views for sharing
- Import_shares - view inbound and outbound shares, create databases from a share

DDL commands:

- SHOW Shares command - kind column can specify Inbound shares and outbound shares
- Alter shares sharename add accounts = acctname
- Describe share sharename - details of contents of a share

Stages

Stages can be imagined as temporary locations where data can be stored before it is loaded into databases or unloaded from databases

- Types of Stages - User (@~), Table (@%) and Named stages (@)
- External Stages - Named stages
- External stages can be cloned - do storage integration objects also get cloned?
- Permissions on s3 bucket
    - s3:GetBucketLocation
    - s3:GetObject
    - s3:GetObjectVersion
    - s3:ListBucket
- Internal Stages cannot be cloned.
- Internal / External stages can be defined a s temporary at the time of creation. Stage is dropped at end of session and all files in it are purged.
- Default compression of unloaded files - GZIP (brotli, zstandard, bzip2 also supported)
- Compression for Parquet while unloading to external stage - snappy (default), LZO
- Default file format - CSV, for loading as well as for unloading
- For unloading default format is CSV but JSON and parquet is also supported
- Encoding : default charset : UTF-8
- CSV Gzip is fastest file format for loading
- Internal stage encryption:
    - On server side : AES - 128 bit keys default, can be set to AES-256 also / snowflake_sse
    - On client side : Snowflake_Full (default) /
- Commands-
    - Copy into <table> / <location> needs a warehouse (Bulk loading)
    - Copy_into can copy to internal, external stages and external locations like S3 buckets
    - GET command to download files from internal stage not from external stages to local directory. Use utilities provided by Cloud service to download from external stages.
    - PUT command to put files into stage from location or from file
        - Table stage
        - User stage
        - named internal stage

        - NOT external stage - use cloud utilities here
        - Can be used only thru snowsql
- Directory tables:
    - On both internal and external stages
    - Not a separate object, built implicit object of a stage
    - Does not have grantable privileges
    - use File URLs to access stage files (which does not expire)
    - Need manual data refreshes, but automated refreshes can also be configured thru snowpipe.
    - A small charge is applied for event notifications for refresh of the staged files. Charge appears under snowpipe head
    - Streams can be used with directory tables. Stream is created on the stage object
    - Table functions to retrieve directory table history:
    - AUTO_REFRESH_REGISTRATION_HISTORY
    - STAGE_DIRECTORY_FILE_REGISTRATION_HISTORY
- Details of files staged
    - File URL
    - Relative Path
    - Last modified
    - Size
    - MD5 Checksum
    - eTag

User Stage	Table Stage	Named Internal	NamedExternal
@~	@%, same name as table	@	@
Cannot be altered / dropped	Cannot be altered / dropped	Can be altered / dropped	Can be altered / dropped

Does not support File Format (Specify in Copy into)	Does not support File Format (Specify in Copy into)	Can specify File format at stage level	Can specify File format at stage level
Only current user can access	Multiple users can access	Any user with correct privileges can access	Any user with correct privileges can access
Can update multiple tables with correct privileges	Can update only single table	Can update any table	Can update any table
COPY transformations are possible	Cannot transform data while loading	COPY transformations are possible	COPY transformations are possible
Implicit stage tied to user	Implicit stage tied to table	Separate Database object	Separate Database object
Cannot be cloned	Are cloned, but are empty	Cannot be cloned	Can be cloned
		Can download files to local drives using GET/PUT	Can download / upload files to / from local drives using cloud utilities
No storage or maintenance cost as stage is external	Contribute to storage costs	Contribute to storage costs	No storage or maintenance cost as stage is external
Automatic refresh of metadata	No	No	Yes

Command Options:

- Copy_into *tablename* from *stageName fileformat*=FileFormatname
- GET_STAGE_LOCATION function (@stageName)
    - Returns file URL for a stage, which is permanent
- BUILD_STAGED_FILE_URL function (StageName, Relative file path) -
    - Returns file URL for a stage file, which is permanent
    - URL does not expire
- BUILD_SCOPED_FILE_URL function -
    - URL for files in a stage, valid for 24 hrs / until persisted query result period ends / results cache expires, which is 24 hrs.
    - File can be accessed only by user who generates the URL
    - Allows access of unstructured data in a stage file
    - non-deterministic
- Pre-signed URL:
    - A user who does not have AWS credentials or permission to access an S3 object can be granted temporary access by using a presigned URL. A presigned URL is generated by an AWS user who has access to the object. The generated URL is then given to the unauthorized user.
    - Ideal for BI or reporting tools
    - Configurable access time: expiration_time argument
    - Allows access of unstructured data in a stage file
    - non-deterministic
- LOAD_UNCERTAIN_FILES - To load files whose metadata has expired
- FORCE - to ignore load metadata and load all files
- Overwrite = True
- SINGLE = false - default, download as multiple files. Set to true to download large files as a single file or as smaller files
- Max_File_size - default 16 MB, max is 5GB for cloud , S3, Azure / google storage
- PURGE option to delete files from stage after copy is complete.
- Metadata columns for staged files:
- METADATA$FILENAME - can be queried
- METADATA$File_Row_Number - can be queried
- METADATA$File_CONTENT_KEY
- METADATA$File_LAST_MODIFIED
- METADATA$START_SCAN_TIME
- VALIDATION_MODE <Table> - to get possible errors before query is triggered
    - Return_all_errors

        - return_errors
        - Return_n_rows
    - VALIDATION_MODE <location> -
        - return_errors
    - VALIDATE - to get error history of previously executed queries
        - Takes job_id as parameter
    - Validation_Mode and validate parameters do not support copy statements which transform data
    - List @mystage - lists files that have been staged for further processing
    - Show stages like, in, etc - all stages assigned to the user
    - Remove - removes file from stage

Query Acceleration Service:

- Used when large datasets need to be scanned or filters are to be applied
- Used when need to perform ad hoc analytics, workloads with unpredictable data volume per query, and queries with large scans and selective filters
- Business edition / higher
- Enable_query_acceleration = true
- Query-acceleration_max_scale_factor - default 8 , 0-100, scale factor is the size in times of the VWH that can be deployed when QAS is triggered.

Optimisations

- Query Optimisation services
- Search optimisation service:
    - Enterprise edition or above
    - The search optimization service can significantly improve the performance of certain types of lookup and analytical queries that use an extensive set of predicates for filtering.
    - Queries with IN or equality predicates
    - It can be used in following scenarios:
        - Point lookup queries - return only one or a few rows using highly selective filters.
        - Substring & RegEx searches – queries that use LIKE, ILIKE, & RLIKE
        - Queries on fields in VARIANT, OBJECT & ARRAY columns – using equality conditions, IN, ARRAY_CONTAINS, ARRAY_OVERLAP, Substring & RegEx and NULL check conditions

        - Queries that use specific geospatial functions with GEOGRAPHY values.
    - Not supported on materialized views
    - Not supported on external tables
    - Do not support cast on table columns
    - Warehouse / storage optimisation
- Privileges -
    - To add or create optimisation on a table - ADD SEARCH OPTIMISATION - on schema and ownership on table
    - To use optimisation while running select queries - select privilege on table

Network Policy

- Available for all editions
- Support IPV4 IP addresses
- Can be created for Account, security integrations , user. This is also the order of precedence when there are multiple policies.
- Only one policy for an account / user
- Blocked IP addresses applied first
- Security admin or higher role, or a role with CREATE NETWORK POLICY can create a network policy
- Ownership can be transferred to a role
- A policy comes into effect only after it is activated.
- Policy can be activated by Security admin or higher or with role having global ATTACH_POLICY privilege.
- MINS_TO_BYPASS_NETWORK_POLICY - snowflake support
- Max num of network policies - 50

Object tagging

Allows users to identify data objects that contain sensitive data

- Managed Access Schemas: to centralize privilege management with the schema owner
- Classification
- identify, categorize, and protect sensitive data
- Supports all datatypes except - variant, array,object, vector, geography, binary

Replication

- Available on all editions
- Only ORGAdmin role can enable it
- Data is replicated to atleast 3 availability zones (including primary zone)
- Use SYSTEM$GLOBAL_ACCOUNT_SET_PARAMETER function to enable
- Snowflake provided compute resources are used for replication
- Failover / failback - Business critical and above
- Replication
    - Client Redirect
    - Database Replication
        - Available on all editions
        - Only databases and database objects can be replicated.
        - Cloned database from primary account is copied physically in secondary database. This incurs cost.
        - Historical data (Data in time travel), is not replicated
        - Database replication is possible across accounts, across cloud platforms
        - Replication of materialized views - definition replicated by default, by a background service, data is not refreshed. Automatic maintenance is also not enabled by default.
        - Privileges granted on database objects are not replicated to a secondary database
        - Cannot replicate following database objects:
            - External tables.event tables, Iceberg tables,hybrid tables Replication halts if database contains external table
            - Databases created from shares.
            - Temporary tables.
            - Stages
            - Pipes
    - Account Replication
        - Account objects can be replicated
        - Business critical / higher
        - Grants are not replicated
        - Users and roles are replicated and warehouses are replicated by creating a failover replication group
        - Cannot replicate following objects:

1. Business critical to lower editions (But is possible with IGNORE_EDITION_CHECK clause of ALTER DATABASE)
2. Business critical accounts with signed business associate agreements for PHI data - to Business critical accounts with no such agreement

- Share Replication
  - Business critical / higher
  - Inbound shares cannot be replicated
  - Failover Group
  - Business critical / higher
- Client Redirect
  - Business Critical edition and above
  - Redirect to different region
  - For business continuity, data replication

Billing - Data transfer and compute costs are billed to the target account.

Storage Integration objects

Storage integration object can be imagined as an interface between snowflake and external cloud,storage location (type - external stage)

- First class account level object
- A stage links to a storage integration using a hidden ID
- One stage many Storage-Integration objects
- Can reference multiple external stage locations of the same cloud.
- Params that it takes:
  - Name
  - Type - external stage
  - Cloud
  - Enabled - true / false
  - cloud providerparams
    - Storageprovider - S3 / GCS / Azure
    - AWS_role_ARN - IAM creds / tenant ID /
    - Bucketname
    - ACL

        - STORAGE_ALLOWED_LOCATIONS : parameter to use while creating a STORAGE integration object to limit external stages that use the integration to reference one or more storage locations.
        - Optional - STORAGE_BLOCKED_LOCATIONS : stage locations which cannot be accessed
    - If a storage integration object is changed after it has been linked to one or more stages, you must re-establish the association between each stage and the storage integration by using alter command

Data Loading (Bulk Loading)

- File Formats:
    - Loading: COPY_INTO <Table> supported file formats: CSV, TSV, XML, JSON, AVRO, ORC, Parquet
    - Unloading: COPY_INTO <Location> supported file formats: CSV, JSON, Parquet
    - COPY_INTO <Location> parameter SINGLE = FALSE (default)- copies into multiple files
    - COPY_INTO <Location> : can unload data to internal stage, external stage, external location but not local drive
    - Parquet file - max size allowed - 256 - 512 MB
    - Parquet does not truncate floating point numbers, CSV and JSON truncates
    - When data is unloaded to CSV or JSON - Floating point values are truncated to 15,9
    - Order of precedence - copy into statement -> stage def -> table def
- Compression :
    - Auto-detection - default : gzip, bzip2, deflate, raw-deflate
    - Explicit mention - brotli, zstandard
- Encoding : UTF-8
- Encryption :
    - default 128 bit keys,
    - with additional config 256 bit
    - for already encrypted files: user specified keys
- Loading happens in a single transaction
- Metadata stored for loading into table: eTag, File Name, file size, # rows loaded, timestamp of last load, errors if any during load
- When file load cannot be determined file load is skipped.
- Factors affecting data load:

    - Physical location of stage
    - GZip compression efficiency
    - Number and types of columns
- Copy Command - file parameter - max 1000 files.
- Transformations allowed are omit, cast, column reordering, truncate.
- Transformations which involve processing are not supported for eg: Transpose, Pivot, etc
- Not allowed - Filtering (where, order by, limit, fetch, top), aggregations (flatten, join, group by)
- VALIDATION_MODE :
    - RETURN_<n>_ROWS | RETURN_ERRORS | RETURN_ALL_ERRORS] - tests files for errors but does not load them
- Error notifications:
    - On_Error : abort_statement - notifications are sent. (default)
    - On_Error : Continue - no notifications
    - On_Error : Skip_File - notifications are sent.
    - On_Error : Skip_File <num>- notifications are sent.
    - On_Error : Skip_File <num %> - notifications are sent.

SnowPipe

Snowpipe can be imagined as a service running in the background, which keeps polling stages (internal / external) for loading data into a queue and then snowflake loads data into respective tables.

- Schema level object.
- Named First class schema object.
- Serverless, i.e. it would not need a user defined warehouse for its execution
    - Warehouses will be provisioned in the background whenever snowpipe is needed.
    - Snowflake also gives an option of configuring a warehouse for snowpipe.
- Does not need virtual warehouses. Uses snowflake managed compute resources.
- Copies files from stage to table.
- Uses COPY INTO statement internally: Configured inside a CREATE PIPE statement at the time of creating a pipe. *Syntax: COPY INTO table from stagename*
- Used for transferring small datasets in micro-batches frequently.
- File Size: 100 - 250 MB
- Loads data within a minute.
- Can work with external as well as internal Stages.

- File with same name if loaded earlier is not loaded again, even if data inside it has changed since previous load. But files can be loaded again if FORCE = TRUE
- Loads are combined or split into multiple transactions.
- Snowpipe cannot delete files from stage automatically once copy / loading is completed. They must be removed/purged manually.
- Can be suspended / resumed.
- Pipes referencing internal stages cannot be cloned
- Pipes referencing external stage can be cloned.
- PIPE_EXECUTION_PAUSED - can be set at Acct, schema and pipe level
- PIPE_USAGE_HISTORY table function - for viewing pipe utilization. Takes parameters for date / time range.
- Load history stored in metadata of PIPE for 14 days
- For detecting files available for load in stage
- Automated SnowPipe - auto_ingest = true
- Only external stages
- Cloud messaging notifications
- Snowpipe API endpoints:
    - insertFiles - Returns Success when Snowflake records list of files to add to the table.
    - insertReport - Returns list of files inserted via insertFiles endpoint and data is committed to table for past 10 mins and not more than 10000 records
    - LoadHistoryScan - takes time range parameter to return list of files inserted via insertFiles and data is committed to table
- Works with external as well as internal stages
- Supports key pair auth with JWT.
- Error notifications:
    - On_Error : Skip_File - notifications are sent. (default)
    - On_Error : Continue - no notifications
- DDL commands:
    - CREATE PIPE
    - ALTER PIPE
    - DROP PIPE
    - DESCRIBE PIPE - lists details of a PIPE
    - SHOW PIPES - lists all PIPES for which user has access
- Privileges:
    - Database - USAGE
    - Schema - USAGE, CREATE PIPE
    - External Stage - USAGE

    - Internal Stage - READ
    - Table in PIPE def - SELECT, INSERT
- Billing:
    - Resource Consumption : per-second/per-core granularity
    - Resource consumption of all pipes in an account + 0.06 credits / 1000 files notified, REST API calls or files queued
    - All credits consumed by snowpipe are gathered under warehouse named SNOWPIPE
    - GET
    - PUT

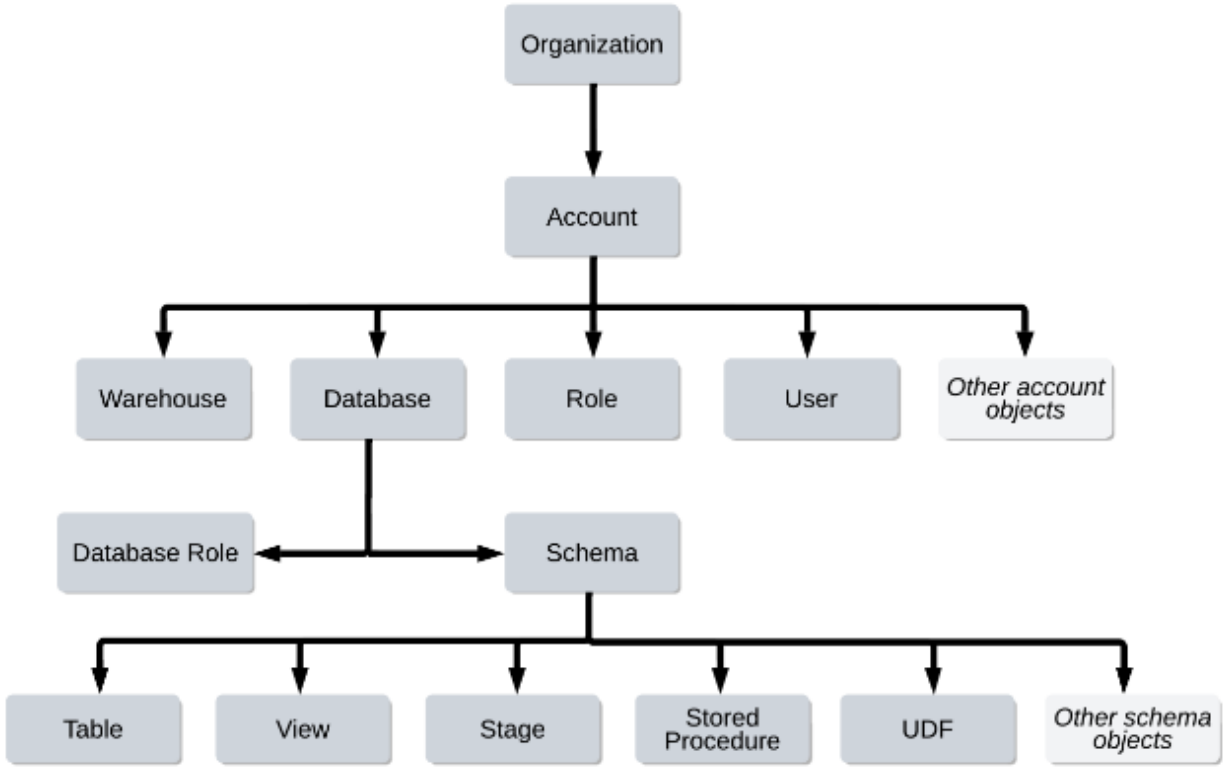
Roles in Snowflake

- OrgAdmin:
    - Everything
    - Rights to set replication
- AccountAdmin:
    - Everything
    - MFA compulsory
- SecurityAdmin:
    - Manage Grants, users and roles
    - Creating network policies
    - Security Admin -> UserAdmin -
        - First user to be created after accountAdmin, so all users and roles can be created with this role.
        - Creating users and roles.
- Sysadmin:
    - Creating warehouses, databases, etc.
    - Custom roles are assigned to this role.
- Public:
    - For each user and role created in snowflake.

Access Control

- DAC - Discretionary Access Control

Each object has an owner which can in turn grant access to other objects (databases, tables, warehouses, roles, users, etc)



- RBAC - Role Based Access Control
  - Access privileges are assigned to roles and roles are assigned to users.
  - Roles are granted privileges to an object using Grant ownership command. Roles are then assigned to users
- Show Grants:
  - SHOW GRANTS OF - Lists users and roles
    - APPLICATION ROLE <Application Name > <application role> (database) ROLE <role name>
    - SHARE <share name>
  - SHOW GRANTS TO - PRIVILEGES granted to
    - <Application Name > (database)
    - <Application Role> (database)
    - <ROLE> (instance role)
    - <SHARE>-

        - <USER>-
    - SHOW GRANTS ON - privileges that have been granted to the object
        - Account <Object name>
    - SHOW GRANTS IN - Lists All privileges
        - Schema <schema name>
        - Database <database name>

Network Policies

- Available for all editions
- Can be created for Account, user / security integrations
- Security admin or above or roles with Global ATTACH POLICY privileges can create a network policy
- Tag based masking
- A tag can support one masking policy per datatype
- Column masking techniques
- Dynamic data masking
- External tokenisation - to tokenise data while data loading and de tokenise at query runtime, i.e. : substitute a randomly generated identifier for sensitive data, in order to prevent unauthorized users access to the data, before loading it into Snowflake

Masking Policies

- Schema level objects
- Enterprise edition or higher
- Types : Dynamic data masking, External tokenisation, tag based, Role Hierarchy, row access policies
- Column and row level security is achieved by masking policies
- Alternative to Secure objects like views, else there can be an explosion of all such objects
- A masking policy consists of a single data type, one or more conditions, and one or more masking functions.

Authentication

- All data is encrypted by default, requires no configuration / management.
- AES-256-bit encryption
- Charge for encryption: free of cost
- Hierarchical key model in hardware security module
- Snowflake managed keys
    - Rotated automatically after 30 days
- Federated authentication - SSO
    - All editions
    - Uses SAML 2.0
    - Separates authentication from access
- MFA - All editions
    - JDBC and ODBC drivers
    - Python connectors
    - Snowsight
    - Snowsql
    - MFA can be cached, ALLOW_CLIENT_MFA_CACHING account parameter, time
    - Mins_TO_BYPASS_MFA - account administrator
    - Cached MFA token valid for upto 4 hours
- Hierarchical Key model:
    - 4 layers of keys
    - Root, Account master keys, table master keys, file keys
    - Root key stored in HSM
    - Rotation - 30 days
    - Periodic Rekeying:
    - Enterprise edition or higher
    - No downtime, data rekeying is a seamless operation
    - Impacts on cost of time travel and fail safe data
- Key rotation
    - Tri Secret Secure Keys:
    - Composite master key : - Snowflake maintained key + Customer managed Key
    - Available on Business critical edition and and up
    - Key pair
    - Kafka, python, spark connectors, snowsql, jdbc, odbc divers use key pair auth
    - VPC authentication
- Client Side Encryption
    - Tri-Secret Secure - Business Edition / VPS

- Query Statement Encryption - Business Critical

SnowPark

- Libraries for
  - Java
  - Python
  - Scala
- Can read data from:
  - Internal stages
  - External stages
  - Snowflake tables

Resource Monitors

Resource Monitors is an arrangement by snowflake to keep credit consumption of virtual warehouses and cloud services under control.

- Virtual warehouses and cloud services which support warehouses can be monitored.
- Available in all editions.
- Can be created by Account admins
- Notifications are sent via email and web console.
- Account monitor notifications are sent by Email only to account admins.
- Non administrators can also receive email notifications but for warehouse level only
- Can be created at account level / at warehouse level (MONITOR_LEVEL).
- Credit Quota, monitor level, schedule and actions are configured.
- Can have multiple triggers
- TRIGGERS ON …. <PERCENT> DO <ACTION>
- Actions - DO SUSPEND | SUSPEND_IMMEDIATE | NOTIFY : Notify and Suspend(1 action per resource monitor) , Notify and Suspend immediately(1), Notify Only (5)
- They can be created at a frequency of yearly, monthly, weekly, daily or never
- A resource monitor can be set to monitor multiple warehouses
- Warehouse can be assigned only to a single resource monitor.
- A Single resource monitor can be assigned at account level
- For associating a warehouse to a RM, RM name is specified in the alter statement of the warehouse under the Tag of RESOURCE_MONITOR
- Resource monitors created for Reader Accounts can be monitored from the RESOURCE_MONITORS view in READER_ACCOUNT_USAGE Schema
- Privileges to non account admins:
- Monitor - viewing of Resource monitors

- Modify - alter RM properties

DDL commands:

- CREATE RESOURCE MONITOR
- SHOW RESOURCE MONITORS
- ALTER RESOURCE MONITOR
- DROP RESOURCE MONITOR

Important miscellaneous functions

Date_trunc - for separating year, mon and date component of date

SET var_name = value

Object_Construct - for converting relational table rows into a variant object

Current_version - snowflake version

Current_client - jdbc / odbc clients version

—----------

Durations

Action	Duration
Load History for bulk data loads - in Metadata of target table	64 days
Load History for SnowPIPE - in Metadata of target table	14 days
Load metadata for files that have been loaded	64 days
Query Result cache	24 hrs, limit reset if cache is queried before 24 hrs and retained for max 31 days
Local / Warehouse cache	14 days
Account_usage - Load_History view	365 days (only 10000 rows, only copy_into statements by bulk load not snow pipe)
Account_USAGE - COPY_HISTORY view	365 days(includes data from snowpipe, no 10000 row limit)
Information_schema - COPY_HISTORY TABLE FUNCTION	14 days
Information_schema - Load_History view	14 days
Account_USAGE - TASK_HISTORY	7 days
Information_schema - Query_History	7 days
Account_usage- Query_History	365 days

insertReport - snowPIPE API	retains data for past 10 mins / 10000 records
Time travel : Standard edition	1 day
Time travel : Enterprise edition and above	0 - 90 days
Time travel : Default standard retention period	1 day (24 hours)
Task runs for a default of	60 mins
Account and Table master keys are rotated	30 days
MAX_DATA_EXTENSION_TIME_IN_DAYS	14 days default and max also
STATEMENT_TIMEOUT_IN_SECONDS - Max time a query will run before getting aborted	48 hrs - default after which it will abort
Max time a query will run in a virtual warehouse, i.e. max value for parameter STATEMENT_TIMEOUT_IN_SECONDS	7 days
STATEMENT_QUEUED_TIMEOUT_IN_SECONDS Max time a query can stay in a queue before it is cancelled	0
Merge Update / Delete queries default lock_timeout	12 Hrs
BUILD_SCOPED_FILE_URL	24 Hrs
Rekeying -	12 months
If the transaction is left open, Snowflake typically rolls back the transaction after it has been idle	4 hrs

Cached MFA token is valid for	4 hrs

Size limits

Loading files from web console	50 MB
Storage layer - microPartitions	50 - 500MB uncompressed
size limit the VARIANT data type imposes on individual rows	16 MB compressed
Bulk load file size : recommended size for data files to optimize the number of parallel operations for a load	100 - 250 MB compressed
Snowpipe recommended file size	100 - 250 MB compressed
Resource consumption of all pipes in an account	0.06 credits / 1000 files notified, REST API calls or files queued
COPY_INTO location - MAX_FILE_SIZE	default 16MB, max - 5GB for cloud storage locations
Maximum single file size (MAX_FILE_SIZE) that is supported for Amazon S3, while unloading the data	5 GB
Copy Command - file parameter - max	1000 files
Account level tasks	10000 max
Child tasks	100
Tree levels	1000 max