https://dpnadqm-gjb50719.snowflakecomputing.com/console/login#/

Cred – PavaniBillapati

Password – Laxmi@1969

AWS - [https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2#](https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2)

Un [pavanibillapati112@gmail.com](mailto:pavanibillapati112@gmail.com)

Password – Bapuji@1964

Azure - <https://portal.azure.com/#home>

Un [pavanibillapati112@gmail.com](mailto:pavanibillapati112@gmail.com)

Password – Bapuji@1964

or

Un- [pbill1@unh.newhaven.edu](mailto:pbill1@unh.newhaven.edu)

Password – Laxmi@1969

**Questions**

1. How do you manage user access and privileges in Snowflake?  
2. What is data sharing in Snowflake and how does it work?  
3. How do you monitor and troubleshoot performance issues in Snowflake?  
4. How do you handle data ingestion and loading in Snowflake?  
5. What are the best practices for optimizing performance in Snowflake?  
6. How would you handle scaling and resizing a warehouse based on workload?  
7. Can you explain how concurrency is managed in Snowflake?  
8. What are Snowflake's security features and how do you ensure data protection?  
9. Can you describe how you would handle data replication or disaster recovery in Snowflake?  
10. What are some common ETL tools that integrate well with Snowflake?

**Snowflake Architecture**

**3 layers –**

**Storage** - data is stored, stored in amazon aws s3 buckets in our case, it can be azure or gcs based on selection in beginning of the account

Data is stored as hybrid columnar storage(saved in blobs) unlike traditional storage like rows – makes more efficient

**Query Processing(Muscle of the system)**

**Virtual warehouses** – virtual computing resources to process queries and other computations

Can perform massive parallel processing

Can be of varying size.

Smallest size is XS consisting of only 1 server. More complex queries are, we can increase the size. Every step we double the number of servers.

For 4XL(largest), consists of 128 servers

**Cloud Services**

Brain of the system

Manage the infrastructure, accesss control, security, optimizer, metadata etc

**Multi clustering**

If there more queries, and if ur account is enterprise level, multiple Clusters can be employed to perform processing instead of queuing them

Warehouses can be created either

by interface

or

by sql commands in worksheet

create or replace warehouse compute\_warehouse

with

warehouse\_size = XSmall

Max\_cluster\_count = 3

MIN\_CLuster\_count = 1

Auto\_suspend = 600

Auto\_resume = TRUE

initially\_suspended = True

comment = "This is second warehouse"

Deleting warehouse - Drop warehouse <warehousename>

**Scaling Policy of clusters**

Dynamically scale depending up on load – auto scaling

Useful if we have more queries(deploy more same size clusters) but not when a single query is complex(increase the size of cluster from S to M or so on)

If we have a single cluster and multiple queries to be processed, a queue is maintained for other queries to be executed.

**2 polices**

**Standard**

Favors starting additional warehouses over conserving credits

Cluster starts when there’s a query in queue or when system detects that there are more queries than can be executed by current clusters

Cluster is shutdown, after 2-3 successful checks(happens every min to see if the least-loaded cluster’s load can be redistributed to other clusters)

**Economy**

Favors conserving credits rather than starting additional warehouses

Cluster starts only if the system estimates that there’s enough query load to keep the cluster busy for atleast 6 min

Cluster is shutdown, after 5-6 successful checks

**Datawarehouse**

**Purpose**

Consolidate and integrate different data sources and use them for reporting and data analysis

We do not want to modify or add extra load on original db’s so, we load(ETL) to a warehouse and do processing there.

We have different layers in between – raw data (extract data from original db’s), staging area

Data integration(transformed and integrated) data transformation

Access layer ( make available for solutions like reporting, data science and other apps)

**Cloud computing**

**Software as a service**

**A computer screen shot of a cloud computing system

Description automatically generated**

**Snowflake Editions**

1. **Stndard**

Introductory level

1. **Enterprise –** additional features in large companies
2. **Business critical** – even high levels of protection if u r dealing with sensitive data
3. **Virtual private –** highest level of security

**Pricing**

**Pay only what u need**

Prices depend up on region

1. **Storage**

Monthly storage fees based on avg storage used per month

Cost is calculated after compression

Varies based on different cloud providers

* 1. **On demand storage –** pay only for what you use
  2. **Capacity Storage –** pay only for defined capacity upfront(cheaper)

1. **Computing**

Charged for active warehouses per hr (depends on size of warehouse)

Billed by seconds(min of a min)

Charged based on snowflake credits

We are charged on credits i.e., we purchase snowflake credits and are consumed

Price also depends on different snowflake editions. For example, for AWS 1 credit in standard edition in us east is $2 where as for Enterprise credit is $3 and business critical is $4 per credit and for virtual private need to contact snowflake

**Usage** –

1. Storage
2. Computing
3. Data transfer – for import u wnt be charged, for exported it wud be

If data is being transferred to other snowflake account within same region using same provider, there is no charge

If the account is in different region or using different provider, then we r charged

**Roles**

Different roles have different privileges

**Predefined roles (5)**

1. Account admin
2. Sys Admin
3. Public
4. User admin
5. Security admin

Additional roles can be created and assigned under sysadmin

A black background with blue text

Description automatically generated

A screenshot of a computer

Description automatically generated

Custom roles can be created & assigned usually to sys admin

Default role assigned to all users is public

**Loading Data**

1. Bulk Loading

We r using compute power of our warehouse using stages

Copy command from sql is used in stages and transformations is possible using this command

1. Continuous loading

For small volumes of data that need to update

A screenshot of a computer screen

Description automatically generated

**Stages**

Not same as DW stages

These are DB objects that contain info about where data has to be loaded from

**Types**

1. **External Stage** – S3, GCP, Azure(external cloud providers)

These are DB objects created in schema using create stage(with url and access details)

Additional costs may apply if region/platform differs

1. **Internal Stage**

Can be a local file destination maintained by snowflake

**Copy**

Load data from stage into databases

Types of loading – Bulk load

Continuous load

For bulk loading, we create stages

Copy command syntax

**On Error**

1. Continue
2. Abort\_Statement
3. Skip\_file

We can specify the limit as well in skip\_file by skip\_file\_limit

Limit here is a number and if file has error rows more than that limit, entire file would be skipped for loading

File format

We can separately create a file format and specify it in copy command file\_format

* Though file format and stage has same properties(stage has few other properties as well other than file format’s), it’s advisable to use file format for setting properities

Copy Options

1. Validation\_mode –

values for this can be (a) return\_n\_rows

(b) return\_errors

- if validation is included in copy command, no rows would be inserted in DB

2. Size\_limit – specify max byte limit that can be loaded as part of the copy command

However even if the limit has exceeded for a file in between, entire file would be loaded.

1. Return\_failed\_only – specifies whether to return only files that have failed to load

-If we use return\_failed\_only it js disaplays errors whether it’s value is set to true or false

But if it’s used along with on\_error = continue option then it dispalys files that are errored out

Or not errored out based on thevalue it is set to.

1. Truncatecolumns – specifies whether the text strings that exceeded the length to be truncated or not

If it is set to True, characters after that target length are automatically truncated

If it is set to false, it produces error if loaded string exceeds the length

Default value is false

1. Force – default is false

If it is set to true, it would load data even though it was loaded and there are no changes after the load, hence creates duplicate data

If it is set to false, it wouldn’t allow to reload same data

**Load history**

Enables you to retrieve history of data loaded into tables using copy into table command

It can be viewed from your database ->information schema ->views -> load history

Or we can also view it from snowflake’s global history where it maintains history across different db’s

Snowflake ->information schema -> views -> load history

Further filtering can be done on these views using select statements and conditions

**Load unstructured data(json, xml)**

**Steps**

1. **Create stage -** connection to where file is stored
2. **Load raw data** – load raw data into separate table with only 1 column of type variant (can handle unstructured data)
3. **Analyse & parse -**
4. **Flatten & load**

can we use insert instead of copy while copying data from stage in snowflake

In Snowflake, you can use both the **COPY INTO** and **INSERT INTO** statements to load data from a stage into a table, but they serve different purposes and have some key differences:

1. **COPY INTO**:
   * **COPY INTO** is typically used for bulk loading data from a stage into a table. It's efficient for loading large volumes of data, and it supports various file formats, including CSV, JSON, Parquet, and more.
   * It's the preferred method for efficiently ingesting large amounts of data, especially when loading data from cloud storage or other external sources.
   * **COPY INTO** allows you to define the file format, file pattern, and other parameters for loading data in a batch process.

**INSERT INTO**:

* **INSERT INTO** is typically used for inserting individual rows or small batches of rows into a table. It's suitable for transactional operations.
* You can use **INSERT INTO** when you want to insert data row by row or when you need to transform data before inserting it into a table.
* It's not as efficient as **COPY INTO** for bulk data loading.

In summary, while both **COPY INTO** and **INSERT INTO** can be used to move data from a stage into a table, you should choose the appropriate statement based on your use case. If you need to load large volumes of data in an efficient manner, **COPY INTO** is the better choice. If you're working with small amounts of data or need to perform additional transformations during the insertion process, **INSERT INTO** is more suitable.

**Optimization**

**Traditional way of optimization**

* Add indexes, primary keys
* Create table partitions
* Analyze query execution table plan
* Remove unnecessary full table scans

All of these & more are automatically managed by snowflake

* Automatically manages micro-partitions

Still we need to take care of

– assigning appropriate data types

----Sizing virtual warehouses

----Cluster keys

---chose right size of warehouse

**Our job**

1. Create dedicated warehouse for different user groups and kind of work (based on their needs and queries)

* Identify & classify user groups or workloads
* Create dedicated warehouse for each group
* Avoid creating too many virtual warehouses for small groups becos though we set auto suspend we might not be using the warehouse and still pay for the warehouse being active
* Refine classifications – workload can change after a certain time.

create role data\_scientist

create role database\_administrator

grant usage on warehouse ds\_wh to data\_scientist

grant usage on warehouse dba\_wh to database\_administrator

create user ds1

password = ps1,

LOGIN\_NAME = dsus1,

default\_role = data\_scientist,

default\_warehouse = ds\_wh

must\_change\_password = false

create user us2

password = ps2

login\_name = dba\_us1

default\_role = database\_administrator

default\_warehouse = dba\_wh

must\_change\_password = false

grant role data\_scientist to user ds1

grant role database\_administrator to user us2

1. Scaling up - when we know, that during certain times load would be higher increase the size of warehouse

-used when query is complex but not more num of users accessing warehouse incase of more concurrent users or more concurrent queries scaling out is done

Increase size of cluster is nothing but increasing from XS -> S or S->M and so on

1. Scaling out – instead of increasing warehouse size, we are introducing multi cluster warehouse when we do not know the patterns of increase in load

If we have enterprise edition, set all warehouses to be multiclusters

Min size – 1 and max size can be high based on need

1. Maximize cache usage – automatic caching can be maximized

Cache – automatically enabled to speedup the execution of queries

i.e., if a query is executed, results are stored in cache and can be reused however results are cached only for 24 hrs or until underlying data is changed.

Cache is automactically enabled, so what can we do ? we need to make sure similar queries go on same warehouse so that we use cache effectively.

1. Cluster keys – for large tables

Tables are divided into partitions and stored

On what basis clusters are created?

* Columns that are commonly used in where clauses – e.g. date columns in event table
* If we use multiple columns, table can benefit from multiple cluster keys
* Columns that are frequently used in joins
* Large num of distinct values for a column or small num of distinct values for a column can be avoided for cluster keys

A screenshot of a computer program

Description automatically generated

**Azure/AWS/GCP**

* + 1. Create IAM role
    2. Create storage integration object
    3. Update policy in cloud
    4. Create file format
    5. Create stage and check if file exists
    6. Create table
    7. Copy data from staging object to table

**Copy data from snowflake to cloud(Unloading)**

Make sure u have enuf privileges i.e., write privilege to cloud and appropriate role (storage admin for cloud)

* Create storage integration object
* Create file format
* Create stage and update url to your destination folder
* Copy data from staging object to table

A screen shot of a computer program

Description automatically generated

**SnowPipe**

* Automatically enables loading once a file appears in bucket
* Snowflake will use and manage their own warehouse for the pipe objects and our virtual warehouses are not used for that
* snowpipe is not intented for batch loading but for continous loading. For batch loading we should just use the COPY command.
* Uses serverless features instead of warehouses

Snowpipe for AWS

A diagram of a serverless load

Description automatically generated

If a new file is loaded to s3 bucket, a notification is made to be triggered which initializes the snowpipe to perform serverless computations and store it into snowflake DB

**Steps**

1. Create stage(file format,storage integration creations) – to have a connection
2. Copy command
3. Create pipe – contains copy def
4. Set up S3 notification – to trigger snowpipe

**Alter pipes**

* 1st pause pipe
* And then makes changes to pipe
* If there are existing files in cloud those has to be manually copied
* Unpause the pipe

**Snowpipe for Azure**

When new file is added to azure containers, event notification would be triggered and placed in Queue storage(type of storage that can store msgs or notifications) and are consumed by snowflake’s notification integration. Which initiates severless load for new notifications and load them to snowflake

**List**

1. Storage integration – connection to container & grant permissions
2. Create stage
3. Queue + notification – to trigger snowpipe
4. Notification integration – notification can be received by snowflake, grant permissions
5. Create pipe

**Time travel**

* Is a feature in snowflake that can be retrieve data that has been deleted by accident
* In standard edition, we can travel upto 1 day
* In enterprise edition/business critical/virtual Private we can travel upto 90 days

Though higher editions, time travel period can go up to 90, it has to be done manually. By default, retention period for objects is set 1, which can be changed upto 90 in higher editions

* Both time travel and undrop are based on retention period field, if it is set 0 data cant be retained.
* Retention period can set either while creating the table itself or it can be alter later

alter table OUR\_FIRST\_DB.public.test

set data\_retention\_time\_in\_days = 2

create table test3(

name varchar(10))

data\_retention\_time\_in\_days = 3

* We can check the amount of data stored and time travelled data stored in snowflake.account\_usuage\_table\_storage\_metrics & snowflake.account\_usuage.storage\_usuage

**3 methods**

* select \* from OUR\_FIRST\_DB.public.test at (offset => -60 \* 20)
* select \* from OUR\_FIRST\_DB.public.test before (timestamp => '2023-11-06 14:15:03.555'::timestamp)
* select \* from OUR\_FIRST\_DB.public.test before (statement => '01b027fb-0504-d116-0001-ca7700019512')

**Restoring table**

**Two ways**

1. Bad way – can use time travel methods along with create or replace the original table

It is bad becos there may be other updates made and since we are replacing the original table we wnt be able to go back to the previous wrong update made.

1. Good way –
2. create another table with time travel select query
3. truncate the rows of original table
4. copy/insert the temp table contents to original table

In this case, we are not replacing the original table hence we can go back to previous timestamps as well.

**Undrop**

We might sometimes accidently drop tables, schemas or even databases, which in snowflake can easily be restored by ‘undrop’ command

**Syntax :** undrop table/schema/database <it’s name>

**Restoring replaced table**

Table which has been replaced by using create or replace can also be stored using undrop

1. 1st the replaced table name has to be updated using alter command
2. Then undrop the table

Now we can perform time travel done even before replace or the table

**Fail Safe**

* Data protection in case of disaster
* We get non-configurable 7 day period for permanent tables
* This period starts after retention period
* We cant recover data from fail safe by ourselves need to contact snowflake service directly
* Storage costs incur even for fail safe data stored

A screen shot of a computer

Description automatically generated

**Table types**

1. Permanant table
2. Transient Table
3. Temporary table – available only for session/user, if it’s closed wnt be available and no time travel as well

**A screenshot of a computer

Description automatically generated**

For managing storage costs we have different types of tables

* These types are available for other database objects as well.
* There wont be issue if we have same name for temporary table and permenant table
* For transient tables, we wnt be able to set the retention period other than 0 or 1. If we set to 0, we wnt be able to time travel. And no fail safe
* We can check the type of table using show databases and options column – blank if it’s permenant or show othertype of table

Or

* We can also check in snowflake.account\_usuage.table\_storage\_metrics – is transient column

Or

* We can also see using show tables in kind column
* A temporary temple is only existing within one session - yet a transient table can exist beyond the session

**Zero-copy cloning**

Easy to copy table with it’s metadata and structure with clone command & improved storage

Copied table is independent of original table

Creating backups

Works with time travel also

**Syntax** – create table <t> clone <source table>

create table <t> clone <source table> before (timestamp => <timestamp>)

A black and white text on a black background

Description automatically generated

* Temporary tables can’t be cloned but however even if the cloned table is temporary it do works

**Swapping tables**

Usecase – for moving development table to production

Many ways – we can create a clone and move it to prod but there’s an easy way of swapping is js to redirect metadata copies

A diagram of a swapping tables

Description automatically generated

Alter table <table name> swap with <target table name>

Alter table <schema name> swap with <target schema name>

For databases, schemas, and tables, a clone does not contribute to the overall data storage for the object until operations are performed on the clone that modify existing data or add new data.

An object clone inherits any metadata, such as comments or table clustering keys.

**Data Sharing**

* Can be done without actually copying data and data would be up-to-date.
* We can share this data with other snowflake users. The former is called producer and later is called consumer. The producer has full access whether the consumer can view or can stop sharing at any point of time.
* The data sharing occurs on consumer computing resources only
* We can share data with non-snowflake users as well js by giving read-only access

Ways – through interface

Through SQL

Check the shared data from consumer account using these

A screenshot of a computer

Description automatically generated

**Sharing data with non-snowflake users**

Steps

1. Create a reader instance in our account – is an independent instance with own url and compute resources (done in our account)
2. Create share database and table (done in our account)
3. In reader account, create database from share
4. We can create multiple users with in the reader account based on requirement

* If u r not seeing any shares when show shares is executed, check if u r logged in as account admin
* If we r creating users, these are assigned to public

**Sharing databases & schemas**

* Same steps instead of grant share on single table we would grant access to all tables in db or schema
* If we are makings any changes to the table after share, the makes would be reflected
* However if new table is created it wont be visible

**Views & Secure Views**

* Views sharing displays what data is being shared i.e., the view creation sql command is displayed & when we do not want to do so we use secure views
* Only secure views can be shared
* Secure view can be created just by adding secure before view in view creation statement

**syntax**

Create or replace secure view <view name> as (select statement)

**Data Sampling**

Data sampling is the process of selecting a representative subset of data from a larger dataset for analysis

Snowflake provides several methods for sampling data, and the approach you choose depends on your specific requirements and the characteristics of your data.

Top of Form

Sampling using the SYSTEM method is faster than the row method

Bottom of Form

A black and white screen with white text

Description automatically generated

**Tasks**

Tasks are objects that are used schedule SQL statements

Tasks can be standalone or tree of tasks i.e., we can create dependent sql statements

**Tree Tasks**

There can be multiple sub tasks under a parent task however there can be a child task can have only one parent task

A screenshot of a computer program

Description automatically generated

We should resume child tasks 1st

**Conditional tasks**

We can include condition in tasks & when that condition is evaluated to true task would execute

**Syntax** create or replace task <taskname>

warehouse = <wh name>

schedule = ‘1 minute’

when <condition>

as

<insert statement>

**Streams**

Datawarehouses is built by performing ETL operations & retrieving data from different sources.

When there are some additional rows inserted in these sources, we want only these updations in warehouse or when some existing columns are updated r deleted we want these updations to be maintained even in warehouse. This is exactly what streams are used for.

Streams are objects that record changes(DML – insert update & delete) made to the table. This process is called change data capture(CDC)

Stream object is created on table. If any create, update or delete operations are performed on table, stream objects stores these changes with all table columns and 3 additonal rows – metadata$action, metadata$update, metadata$row\_id

Stream objects doesn’t store table’s original columns again so there are no additional storage costs except for metadata columns storage

A screenshot of a stream

Description automatically generated

The stream object table contents are deleted after the warehouse has been updated with the new changes in table

Metadata$action – contains operation performed

Metadata$update – contains true/false based on whether it’s an update is performed on table

For update operation, there would be two rows inserted in the stream object, for deletion of the row and other row inserted for new data

**Types of streams**

1. Standard - captures insert, updates & delete changes made on table (default if no type is specified )
2. Append only – captures only insert changes made on table

A screen shot of a computer code

Description automatically generated

Alternative method to track changes made on table

**Change Clause**

* For using this on a particular table we need to update change\_tracking to True on that table using alter command
* We can check the changes made using select statement with changes(information => ‘<value>’) with time travel features offset & timestamp
* This value can be either

1. Default – captures all changes like insert, update & delete
2. Append only - Append-only streams track only inserts. Update and delete operations are not tracked.

* If we are updating the same row, only the latest change would be available
* One advantage with change clause is even after it is consumed we can still access those changes unlike streams

**Materialized views**

Views are js stored query & is re-runned every it’s called. It would be time-consuming if these queries are complex.

Materialized views stores results of the view js as a table so it’s fast when we perform on any operations on it.

If any changes occur to the data in original table, we expect those changes to be reflected, it is automatically handled by snowflake

Since it’s handled by snowflake, the computational costs doesn’t in occur on warehouse but we need to pay additionally for this service.

We can check it at when logged in as account admin & under admin – usage can check for computational costs – materialized views costs

**When do we use materialized views?**

**A screenshot of a computer

Description automatically generated**

If the data is updated on regular basis, we can use streams & tasks as an alternative

If there are any changes in table, we can set streams to update using task with merge

A black background with white text

Description automatically generated

**Limitations**

* Only enterprise or higher editions have this feature
* A screenshot of a black and white message

  Description automatically generated

UDF (user defined functions)

**Data Masking**

Data masking is a technique used to protect sensitive information by replacing, encrypting, or scrambling original data with fictional or pseudonymous values. The primary goal of data masking is to ensure that sensitive information remains confidential

Data masking in snowflake is done at column level

We can customize data masking i.e., specify hw many characters can be made visible

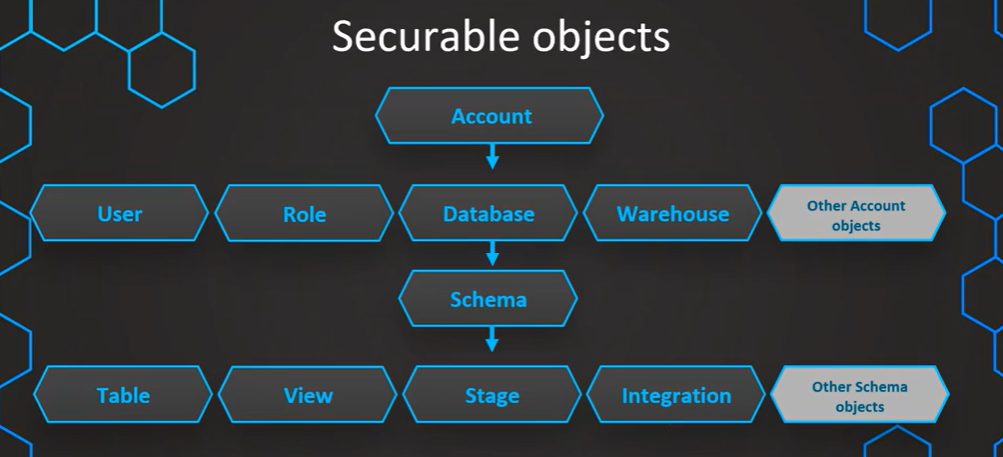
**Access Control**

1. **Discretionary access control**
2. **Role based access control**

**A screenshot of a computer

Description automatically generated**

If u have created an object u have full access to it & can grant access to different roles. These roles in turn are assigned to users who can therefore access



Every object is owned by a single role(multiple users)

A screenshot of a computer

Description automatically generated

**Account admin**

Mainly for initial setup

1st user in system by default becomes account admin

If we create objects this role we need to manually give permissions to these objects for other roles to access them

A screenshot of a computer

Description automatically generated

In snowflake, only account admin & security admin can view account tab. However security admin till have limited access like he can’t see usage, billing r create reader accounts

As mentioned, its imp to setup multifactor authentication to account admin, it can be done by going to account profile -> preferences -> multifactor authentication – enroll in mFA – setup duo

**Security Admin**

Limited access to account admin

**A screenshot of a computer

Description automatically generated**

**Sysadmin**

As custom roles are assigned to sys admin, sys admin can manage r access them

**A screenshot of a computer

Description automatically generated**

If custom roles are not assigned to sys admin, & access sysadmin grants permissions to custom role, sysadmin no longer has access to it & can’t manage this or access objects created by these roles is why we try to assign all custom roles to sysadmin

**Custom roles**

**A diagram of a company

Description automatically generated**

**User Admin**

Unlike security admin, user admin cant create privileges to global objects unless the objects are created by user admin itself

**A diagram of a company

Description automatically generated**

User admin – main purpose - create role & users

**Public**

**A diagram of a company

Description automatically generated**

Objects created using public role can be accessed by all roles

If we have multiple roles assigned to our user we can always change the role using the interface or by the SQL command "USE ROLE

**Best Practice:**

**Virtual Warehouse**

* Always auto-suspend warehouse
* Auto-enable if we are using auto suspend, because there might users trying to execute queries using these suspended warehouses.
* Set appropriate timeouts

A screenshot of a video game

Description automatically generated

**Table design**

* Choose appropriate table type

A black background with white text

Description automatically generated

* Appropriate data type
* Use cluster keys if necessaryA screenshot of a black screen

  Description automatically generated

**Monitoring**

As we pay on usage, we need to always monitor n delete unused tables r warehouses

**Retention period**

Not all tables required same retention period based on usage

A screen shot of a black screen

Description automatically generated

If noticed the last 1, there time travel n fail safe is costing more than the actual table, in these cases we can use transient tables however there’s won’t any backup of data