**Views - Non Materilized views**

It is a Database object which allows the subset of rows or clounms along with dynamic column calculations to be accessed as a table. It stored select statement and its just display the data and it wont store any data. we can defined views as secure or unsecure

A View is a saved SQL query that presents data from one or more tables.

Logical only → does not store data, just the query definition.

Each time you query the view → Snowflake fetches fresh data from the base tables.

Use case: security (mask sensitive columns), simplifying complex joins, or creating reporting layers.

show tables ;

show views ;

select \* from client ;

create or replace view v\_active\_client

as

select \* from client where status = 'ACTIVE' ; -- subset of rows

select \* from v\_active\_client;

create or replace view v\_client as select id, first\_name from client ; --only few columns

select \* from v\_client;

create or replace view v\_client\_extra\_cols as select cl.first\_name ||' '|| cl.last\_name as full\_name, cl.\* from client cl ;

select \* from v\_client\_extra\_cols;

--If there is a large number of clients. Different Views can be created to modularize the data

create or replace view v\_client\_active as select \* from client where status = 'ACTIVE' ;

create or replace view v\_client\_inactive as select \* from client where status = 'INACTIVE' ;

--Can be defined as secure as well

create or replace secure view v\_cleint\_sec as select \* from client ;

-- As we have seen in the last session that when you create a secure view, the body is not accessible to user it is shared

show views ;

select get\_ddl('view', 'v\_cleint\_sec');

grant all privileges on view v\_cleint\_sec to role developer;

grant all privileges on view v\_active\_client to role developer;

------------------------------------------------------------------------------

**Materilized Views:**

-- will stores the data like a table and its behaves likes a table, faster execution of a freqently needed complicated queries.

-- Its uses select statements and charges associated with the kind of views and we can defined as secure or unsecure materialized views.

Limitations: Some of the aggregrated functions, joins, self joins not allowed in the MV's and NO UDF'S, NO LIMIT CLAUSE, and NO HAVING CLAUSE

-- to check the performance of a materialized biew we will disable the caching

alter session set USE\_CACHED\_RESULT =FALSE ;

create table customer\_address as select \* from snowflake\_sample\_data.tpcds\_sf100tcl.customer\_address ; -- takes 28 seconds

-- Lets create a little analytical OR complicated columns query

-- which might take some extra time when doing it at run time

-- We will add some max, min, -- some aggregate functions allowed in Materialized views

select ca\_address\_id, count(distinct ca\_zip) as num\_zips, avg(ca\_gmt\_offset) as avg\_gmt\_offset, count(\*) num\_address, max(ca\_suite\_number) max\_suite, min(ca\_location\_type) min\_loc, sum(ca\_gmt\_offset) sum\_g1\_offset

from customer\_address group by ca\_address\_id

;

-- The above query is taking approximately 35 seconds

-- Lets create a materialized view

create or replace materialized view mv\_customer\_address

as

select ca\_address\_id,

--count(distinct ca\_zip) as num\_zips1

approx\_count\_distinct(distinct ca\_zip) as num\_zips,

avg(ca\_gmt\_offset) as avg\_gmt\_offset,

count(\*) num\_address,

max(ca\_suite\_number) max\_suite,

min(ca\_location\_type) min\_loc,

sum(ca\_gmt\_offset) sum\_g1\_offset

from customer\_address group by ca\_address\_id ;

create or replace secure materialized view mv\_customer\_address\_sec

as

select ca\_address\_id,

--count(distinct ca\_zip) as num\_zips1

approx\_count\_distinct(distinct ca\_zip) as num\_zips,

avg(ca\_gmt\_offset) as avg\_gmt\_offset,

count(\*) num\_address,

max(ca\_suite\_number) max\_suite,

min(ca\_location\_type) min\_loc,

sum(ca\_gmt\_offset) sum\_g1\_offset

from customer\_address group by ca\_address\_id ;

-- understand that all aggregate functions are not allowed. Hence I have to comment the count(distinct)

-- Lets select from the view

alter warehouse compute\_wh suspend;

show views like 'mv%';

select \* from mv\_customer\_address ; -- came back in ~ 19-25 seconds versus ~35 seconds above -- So if you run this frequently this is more efficient.

-- let see materialized views show materialized views

-- check refreshed on, behind by - 0 means latest

--Let’s check the underlying table changes for data --Do some DML – delete and updates and inserts

select \* from customer\_address where ca\_address\_id in ('AAAAAAAAAAFGGJCA',

'AAAAAAAAOKEHALBA',

'AAAAAAAAMAAIGJCA',

'AAAAAAAAMHMHALBA',

'AAAAAAAABOGBDPAA', 'AAAAAAAAHGNNPLAA') ;

update customer\_address set ca\_suite\_number = 'ca\_sn' where ca\_address\_id in ('AAAAAAAAAAFGGJCA',

'AAAAAAAAOKEHALBA',

'AAAAAAAAMAAIGJCA',

'AAAAAAAAMHMHALBA',

'AAAAAAAABOGBDPAA',

'AAAAAAAAHGNNPLAA') ;

-- Let's check the query it should show the data

select ca\_address\_id, count(distinct ca\_zip) as num\_zips, avg(ca\_gmt\_offset) as avg\_gmt\_offset, count(\*) num\_address, max(ca\_suite\_number) max\_suite, min(ca\_location\_type) min\_loc, sum(ca\_gmt\_offset) sum\_g1\_offset

from customer\_address where ca\_address\_id in ('AAAAAAAAAAFGGJCA',

'AAAAAAAAOKEHALBA',

'AAAAAAAAMAAIGJCA',

'AAAAAAAAMHMHALBA',

'AAAAAAAABOGBDPAA', 'AAAAAAAAHGNNPLAA') group by ca\_address\_id

;

-- Check the behind\_by -- how far the materialized is behind the updates in the base table

show materialized views;

--check compacted\_on -- what time the materialized view was compacted

-- Select from materialized view will refresh the view

select \* from mv\_customer\_address where ca\_address\_id in ('AAAAAAAAAAFGGJCA',

'AAAAAAAAOKEHALBA',

'AAAAAAAAMAAIGJCA',

'AAAAAAAAMHMHALBA',

'AAAAAAAABOGBDPAA',

'AAAAAAAAHGNNPLAA') ;

-- check the refresh and compacted by again show materialized views;

-- First select from MV post DML will be a little longer since it refreshes the view. Subsequent runs will be quicker.

-- MV is managed by snowflake compute power

-- It doesn't use the Warehouse

-- You can check the cost of maintenance for materialized view

--

select \* from table(information\_schema.materialized\_view\_refresh\_history()) ;

-- Not a best case if underlying data is changing too frequently

-- materialized view can be suspended

--suspend MV

alter materialized view mv\_customer\_address suspend ;

show materialized views ; --select from MV

select \* from mv\_customer\_address where ca\_address\_id in ('AAAAAAAAAAFGGJCA',

'AAAAAAAAOKEHALBA',

'AAAAAAAAMAAIGJCA',

'AAAAAAAAMHMHALBA',

'AAAAAAAABOGBDPAA', 'AAAAAAAAHGNNPLAA');

--resume

alter materialized view mv\_customer\_address resume ;

--

show materialized views ; -- check the data now

select \* from mv\_customer\_address ; -- materialized view can be clustered like a table can be clustered show materialized views ;

alter materialized view mv\_customer\_address cluster by (ca\_address\_id) ;

-- remove clustering

alter materialized view mv\_customer\_address drop clustering key ;

--- cleanup drop table customer\_address ;

drop materialized view mv\_customer\_address ;