**-----------------------SNOWFLAKE UNMANAGED ICEBERG TABLES ---------------------**

**External volume creation**

-- Snowflake uses an external volume to establish a connection with your cloud storage

in order to access Iceberg metadata and Parquet table data.

CREATE OR REPLACE EXTERNAL VOLUME snowlake\_exvol

STORAGE\_LOCATIONS =

(

(

NAME = 'my-s3-us-east-1'

STORAGE\_PROVIDER = 'S3'

STORAGE\_BASE\_URL = 's3://snowlake-bucket/sales\_data/'

STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::745882220951:role/snowflake\_role1'

)

);

-- Describe the external volume to generate ID and that external id must include in the above role

DESC EXTERNAL VOLUME SNOWLAKE\_EXVOL;

**Creation of the catalog integration**

-- To create an Iceberg table that is not managed by Snowflake, you must specify a catalog integration.

-- A catalog integration is an account-level Snowflake object that defines the source of metadata and Schema for Iceberg tables.

CREATE OR REPLACE CATALOG INTEGRATION snowlake\_glueCatalog

CATALOG\_SOURCE=GLUE

CATALOG\_NAMESPACE='snowlake\_db'

TABLE\_FORMAT=ICEBERG

GLUE\_CATALOG\_ID='745882220951'

GLUE\_AWS\_ROLE\_ARN= 'arn:aws:iam::745882220951:role/service-role/AWSGlueServiceRole-snowlake'

GLUE\_REGION='us-east-1'

ENABLED=TRUE;

-- Describe the catalog integration to generate ID and that id must include in the above role

DESC CATALOG INTEGRATION snowlake\_glueCatalog;

-- Shows the all catalog integration in the snowflake

SHOW CATALOG INTEGRATIONS;

-- By using the crawler, the metadata tables are created in glue catalog, so in Athena we insert table data from glue catalog

--Go to Athena to create unmanaged iceberg table,

-- create table in Athena for customer ,store\_returns ,reason

-- created iceberg table using customer table

create table iceberg\_customer(

c\_customer\_sk decimal,

c\_customer\_id string ,

c\_current\_cdemo\_sk decimal,

c\_current\_hdemo\_sk decimal,

c\_current\_addr\_sk decimal,

c\_first\_shipto\_date\_sk decimal,

c\_first\_sales\_date\_sk decimal,

c\_salutation string,

c\_first\_name string,

c\_last\_name string,

c\_preferred\_cust\_flag string,

c\_birth\_day decimal,

c\_birth\_month decimal,

c\_birth\_year decimal,

c\_birth\_country string,

c\_login string,

c\_email\_address string,

c\_last\_review\_date string)

LOCATION 's3://snowlake-bucket/salesdata/iceberg\_customer' tblproperties('table\_type'='ICEBERG')

-- inserting data from customer table

insert into iceberg\_customer (select \* from customer)

--creation of unmanaged iceberg table in snowflake

-The catalog table name as well as iceberg table name in snowflake should be same as the iceberg table that is created in Athena.

CREATE or REPLACE ICEBERG TABLE iceberg\_store\_returns

CATALOG\_TABLE\_NAME='iceberg\_store\_returns'

CATALOG='snowlake\_glueCatalog'

EXTERNAL\_VOLUME='SNOWLAKE\_EXVOL';

-- Getting iceberg\_store\_returns data from iceberg table

select \* from iceberg\_store\_returns limit 1000;

-- Creating customers iceberg table

CREATE or REPLACE ICEBERG TABLE iceberg\_customer

CATALOG\_TABLE\_NAME='iceberg\_customer'

CATALOG='snowlake\_glueCatalog'

EXTERNAL\_VOLUME='SNOWLAKE\_EXVOL';

select \* from iceberg\_customer LIMIT 10;

**Schema Evolution**

-- Go to Athena to perform **schema evolution**

ALTER TABLE my\_db.my\_table RENAME TO my\_db2.my\_table2

ALTER TABLE iceberg\_customer ADD COLUMNS (comment string)

ALTER TABLE iceberg\_customer CHANGE comment blog\_comment string AFTER id

ALTER TABLE iceberg\_customer DROP COLUMN comment

insert into iceberg\_customer(c\_customer\_sk) values (12112211)

delete from iceberg\_customer where c\_customer\_sk=12112211

--After schema evolution, we need to refresh in snowflake to see changes

--Go to snowflake refresh the unmanaged iceberg table

Alter iceberg table iceberg\_customer refresh;

-- iceberg tables can be DROP and UNDROP

Drop iceberg table iceberg\_customer;

UnDrop iceberg table iceberg\_customer;

--The location of the metadata file and status of the snapshot generation

Select SYSTEM$GET\_ICEBERG\_TABLE\_INFORMATION('iceberg\_store\_returns');

**Creating Shares**

-- Creating a share

Create or replace share CUSTOMER\_DATA\_SHARE

--Creating a secure view ICEBERG TABLE

create or replace secure view SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_CUSTOMER\_VIEW as (select C\_CUSTOMER\_ID, C\_CUSTOMER\_SK, C\_FIRST\_NAME, C\_LAST\_NAME, C\_BIRTH\_COUNTRY, C\_EMAIL\_ADDRESS from snowlake\_db.snowlake\_unmanaged.iceberg\_customer);

-- Granting permissions for secure view

GRANT USAGE ON DATABASE SNOWLAKE\_SHARE\_DB TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT USAGE ON SCHEMA SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE TO SHARE CUSTOMER\_DATA\_SHARE;

grant reference\_usage on database SNOWLAKE\_DB to share CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_CUSTOMER\_VIEW TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_CUSTOMER\_VIEW TO SHARE CUSTOMER\_DATA\_SHARE;

ALTER SHARE CUSTOMER\_DATA\_SHARE ADD ACCOUNT = JZB03580;

-- Creating secure view with two iceberg tables join

--Join the customer table with the sales return table to identify high-value customers who have a history of returns

create or replace secure view iceberg\_customer\_store\_view as

(SELECT C.C\_CUSTOMER\_ID, C.C\_FIRST\_NAME, C.C\_LAST\_NAME, COUNT(R.SR\_ITEM\_SK) AS TOTAL\_RETURNS

FROM iceberg\_customer c

LEFT JOIN iceberg\_store\_returns R ON C.C\_CUSTOMER\_SK = R.SR\_CUSTOMER\_SK

GROUP BY C.C\_CUSTOMER\_ID, C.C\_FIRST\_NAME, C.C\_LAST\_NAME

HAVING COUNT(R.SR\_ITEM\_SK) > 2

);

-- Created secure view(iceberg\_customer\_store\_view) is shared to (CUSTOMER\_DATA\_SHARE)

GRANT USAGE ON DATABASE SNOWLAKE\_SHARE\_DB TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT USAGE ON SCHEMA SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE TO SHARE CUSTOMER\_DATA\_SHARE;

grant reference\_usage on database SNOWLAKE\_DB to share CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.iceberg\_customer\_store\_view TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.iceberg\_customer\_store\_view TO SHARE CUSTOMER\_DATA\_SHARE;

ALTER SHARE CUSTOMER\_DATA\_SHARE ADD ACCOUNT = JZB03580;

-- How to see share objects

SHOW SHARES;

-- How to see the grants of a share object

SHOW GRANTS TO SHARE CUSTOMER\_DATA\_SHARE;

-- CREATING SECURE VIEW WITH ICEBERG AND NATIVE TABLE JOIN

create or replace secure view item\_store\_view as

(SELECT

I.I\_ITEM\_ID,

I.I\_ITEM\_DESC,

COUNT(R.SR\_ITEM\_SK) AS RETURN\_COUNT

FROM SNOWLAKE\_DB.SNOWLAKE\_UNMANAGED.native\_item\_table I

LEFT JOIN SNOWLAKE\_DB.SNOWLAKE\_UNMANAGED.iceberg\_store\_returns R ON I.I\_ITEM\_SK = R.SR\_ITEM\_SK

GROUP BY I.I\_ITEM\_ID, I.I\_ITEM\_DESC

ORDER BY RETURN\_COUNT DESC

);

select \* from item\_store\_view limit 10;

-- created secure view(item\_store\_view) is shared to (CUSTOMER\_DATA\_SHARE)

GRANT USAGE ON DATABASE SNOWLAKE\_SHARE\_DB TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT USAGE ON SCHEMA SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE TO SHARE CUSTOMER\_DATA\_SHARE;

grant reference\_usage on database SNOWLAKE\_DB to share CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.item\_store\_view TO SHARE CUSTOMER\_DATA\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.item\_store\_view TO SHARE CUSTOMER\_DATA\_SHARE;

ALTER SHARE CUSTOMER\_DATA\_SHARE ADD ACCOUNT = jzb03580;

**Data masking Policy**

--applying column level masking policy to unmanaged iceberg table

create or replace masking policy mask\_policy as (val string) returns string ->

case

when current\_role() in ('ACCOUNTADMIN') then val

else '\*\*\*\*\*\*\*'

end;

alter iceberg table if exists iceberg\_customer modify column C\_EMAIL\_ADDRESS set masking policy mask\_policy;

alter iceberg table if exists iceberg\_customer modify column C\_EMAIL\_ADDRESS unset masking policy;

--Here masking policy applied

select \* from iceberg\_customer limit 100;

**Row access Policy**

--applying row level masking policy to unmanaged iceberg table

CREATE OR REPLACE ROW ACCESS POLICY row\_masking\_policy AS (c\_birth\_country STRING) RETURNS BOOLEAN ->

CASE

WHEN CURRENT\_ROLE() = 'SNOWLAKE\_ARL' AND C\_BIRTH\_COUNTRY= 'BAHRAIN' THEN TRUE

ELSE FALSE

END;

alter iceberg table ICEBERG\_CUSTOMER add row access policy row\_masking\_policy on (c\_birth\_country);

alter iceberg table ICEBERG\_CUSTOMER drop row access policy row\_masking\_policy;

-- Here row access policy applied

select \* from iceberg\_customer limit 100;

**Queries for Performance metrics**

-- to clear cache

ALTER SESSION SET USE\_CACHED\_RESULT = FALSE;

set used\_cache\_result = false;

--performance parity check with these queries

SELECT

C\_FIRST\_NAME,

C\_LAST\_NAME,

C\_BIRTH\_MONTH,

C\_BIRTH\_YEAR

FROM

customer

WHERE

C\_BIRTH\_MONTH = 4

AND C\_BIRTH\_YEAR = 1985;

SELECT

SR\_RETURNED\_DATE\_SK,

SR\_RETURN\_TIME\_SK,

SR\_ITEM\_SK,

SR\_NET\_LOSS

FROM

store\_returns

WHERE

SR\_NET\_LOSS < 1300

ORDER BY

SR\_RETURN\_QUANTITY;

SELECT

C\_FIRST\_NAME,

C\_LAST\_NAME,

C\_EMAIL\_ADDRESS

FROM

customer

WHERE

C\_EMAIL\_ADDRESS LIKE '%.com%'

ORDER BY

C\_FIRST\_NAME;

SELECT

SR\_RETURNED\_DATE\_SK,

SR\_ITEM\_SK,

SR\_RETURN\_QUANTITY

FROM

store\_returns

WHERE

SR\_RETURNED\_DATE\_SK = 2451964

ORDER BY

SR\_RETURN\_QUANTITY ASC;

-------------------------**EXTERNAL TABLE IN SNOWFLAKE**---------------------------

--external stage must be different for each external table

--CREATING STORAGE INTEGRATION

create or replace storage integration snowlake\_integration

type = external\_stage

storage\_provider = s3

storage\_aws\_role\_arn = 'arn:aws:iam::745882220951:role/integration-role'

enabled = true

storage\_allowed\_locations = ('s3://snowlake-bucket/salesdata/store\_returns/');

-- Describe the external volume to generate ID and that external id must include in the above role

--DESCRIBING STORAGE INTEGRATION

    DESC INTEGRATION snowlake\_integration;

--CREATING EXTERNAL STAGE

create or replace stage external\_table\_stage

url='s3://snowlakebucket/salesdata/store\_returns/'

storage\_integration=snowlake\_integration;

--listing all external tables

list @external\_table\_stage;

 --Creating file format

create or replace file format snowlake\_format type='parquet';

 --create external table

create or replace external table snowlake\_ext\_table

with location = @external\_table\_stage

file\_format = snowlake\_format;

select \* from snowlake\_ext\_table limit 10;

--external table for customer data

--CREATING STORAGE INTEGRATION

create or replace storage integration snowlake\_integration\_ext

type = external\_stage

storage\_provider = s3

storage\_aws\_role\_arn = 'arn:aws:iam::745882220951:role/snowlake\_ext'

enabled = true

storage\_allowed\_locations = ('s3://snowlake-bucket/salesdata/customer/');

--DESCRIBING STORAGE INTEGRATION

DESC INTEGRATION snowlake\_integration\_ext;

--CREATING EXTERNAL STAGE

create or replace stage customer\_stage

url='s3://snowlake-bucket/salesdata/customer/'

storage\_integration=snowlake\_integration\_ext;

list @customer\_stage;

--Creating file format

create or replace file format snowlake\_format type='parquet';

--create external table

create or replace external table customer\_ext\_table

with location = @customer\_stage

file\_format = snowlake\_format;

-------------- **SNOWFLAKE MANAGED ICEBERG TABLES** ---------------

**Creating External Volume**

CREATE OR REPLACE EXTERNAL VOLUME snowlake\_managed\_exvol

STORAGE\_LOCATIONS =

  (

     (

        NAME = 'my-s3-us-east-1'

        STORAGE\_PROVIDER = 'S3'

        STORAGE\_BASE\_URL = 's3://snowlake-bucket/managed-iceberg/'

        STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::745882220951:role/snowlake-managed-iceberg-role'

            )

  );

**Creating Storage integration**

CREATE OR REPLACE EXTERNAL VOLUME snowlake\_managed\_integration

    TYPE = external\_stage

    STORAGE\_PROVIDER = s3

    STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::745882220951:role/store-role'

    ENABLED = true

STORAGE\_ALLOWED\_LOCATIONS = ('s3://snowlake-bucket/salesdata/store\_returns')

**Creating an external stage**

CREATE OR REPLACE STAGE external\_stage

URL='s3://snowlake-bucket/salesdata/store\_returns'

STORAGE\_INTEGRATION=snowlake\_managed\_integration;

**Creating an Iceberg Table**

CREATE or REPLACE ICEBERG TABLE iceberg\_managed\_store\_returns

(

SR\_RETURNED\_DATE\_SK decimal,

SR\_RETURN\_TIME\_SK decimal,

SR\_ITEM\_SK decimal,

SR\_CUSTOMER\_SK decimal,

SR\_CDEMO\_SK decimal,

SR\_HDEMO\_SK decimal,

SR\_ADDR\_SK decimal,

SR\_STORE\_SK decimal,

SR\_REASON\_SK decimal,

SR\_TICKET\_NUMBER decimal,

SR\_RETURN\_QUANTITY decimal,

SR\_RETURN\_AMT decimal,

SR\_RETURN\_TAX decimal,

SR\_RETURN\_AMT\_INC\_TAX decimal,

SR\_FEE decimal,

SR\_RETURN\_SHIP\_COST decimal,

SR\_REFUNDED\_CASH decimal,

SR\_REVERSED\_CHARGE decimal,

SR\_STORE\_CREDIT decimal,

SR\_NET\_LOSS decimal

)

CATALOG='SNOWFLAKE'

EXTERNAL\_VOLUME='snowlake\_managed\_exvol'

BASE\_LOCATION='managed-iceberg/'

as

select

$1:SR\_RETURNED\_DATE\_SK:: decimal as SR\_RETURNED\_DATE\_SK

,$1:SR\_RETURN\_TIME\_SK::decimal as SR\_RETURN\_TIME\_SK

,$1:SR\_ITEM\_SK::decimal as SR\_ITEM\_SK

,$1:SR\_CUSTOMER\_SK::decimal as SR\_CUSTOMER\_SK

,$1:SR\_CDEMO\_SK::decimal as SR\_CDEMO\_SK

,$1:SR\_HDEMO\_SK::decimal as SR\_HDEMO\_SK

,$1:SR\_ADDR\_SK::decimal as SR\_ADDR\_SK

,$1:SR\_STORE\_SK::decimal as SR\_STORE\_SK

,$1:SR\_REASON\_SK::decimal as SR\_REASON\_SK

,$1:SR\_TICKET\_NUMBER::decimal as SR\_TICKET\_NUMBER

,$1:SR\_RETURN\_QUANTITY::decimal as SR\_RETURN\_QUANTITY

,$1:SR\_RETURN\_AMT::decimal as SR\_RETURN\_AMT

,$1:SR\_RETURN\_TAX::decimal as SR\_RETURN\_TAX

,$1:SR\_RETURN\_AMT\_INC\_TAX::decimal as SR\_RETURN\_AMT\_INC\_TAX

,$1:SR\_FEE::decimal as SR\_FEE

,$1:SR\_RETURN\_SHIP\_COST::decimal as SR\_RETURN\_SHIP\_COST

,$1:SR\_REFUNDED\_CASH::decimal as SR\_REFUNDED\_CASH

,$1:SR\_REVERSED\_CHARGE::decimal as SR\_REVERSED\_CHARGE

,$1:SR\_STORE\_CREDIT::decimal as SR\_STORE\_CREDIT

,$1:SR\_NET\_LOSS::decimal as SR\_NET\_LOSS

from @external\_stage (file\_format => 'iceberg\_format');

**CREATING ICEBERG TABLE FOR CUSTOMER DATA**

CREATE OR REPLACE iceberg TABLE managed\_customer

(

    C\_CUSTOMER\_SK number,

    C\_CUSTOMER\_ID varchar,

    C\_CURRENT\_CDEMO\_SK number,

    C\_CURRENT\_HDEMO\_SK number,

    C\_CURRENT\_ADDR\_SK number,

    C\_FIRST\_SHIPTO\_DATE\_SK number,

    C\_FIRST\_SALES\_DATE\_SK number,

    C\_SALUTATION varchar,

    C\_FIRST\_NAME varchar,

    C\_LAST\_NAME varchar,

    C\_PREFERRED\_CUST\_FLAG varchar,

    C\_BIRTH\_DAY number,

    C\_BIRTH\_MONTH number,

    C\_BIRTH\_YEAR number,

    C\_BIRTH\_COUNTRY varchar,

    C\_LOGIN varchar,

    C\_EMAIL\_ADDRESS varchar,

    C\_LAST\_REVIEW\_DATE varchar

)

CATALOG='SNOWFLAKE'

EXTERNAL\_VOLUME='snowlake\_managed\_exvol'

BASE\_LOCATION='managed-iceberg/'

AS

SELECT

$1:"C\_CUSTOMER\_SK"::number as C\_CUSTOMER\_SK

,$1:"C\_CUSTOMER\_ID"::varchar as C\_CUSTOMER\_ID

,$1:"C\_CURRENT\_CDEMO\_SK"::number as C\_CURRENT\_CDEMO\_SK

,$1:"C\_CURRENT\_HDEMO\_SK"::number as C\_CURRENT\_HDEMO\_SK

,$1:"C\_CURRENT\_ADDR\_SK"::number as C\_CURRENT\_ADDR\_SK

,$1:"C\_FIRST\_SHIPTO\_DATE\_SK"::number as C\_FIRST\_SHIPTO\_DATE\_SK

,$1:"C\_FIRST\_SALES\_DATE\_SK"::number as C\_FIRST\_SALES\_DATE\_SK

,$1:"C\_SALUTATION"::varchar as C\_SALUTATION

,$1:"C\_FIRST\_NAME"::varchar as C\_FIRST\_NAME

,$1:"C\_LAST\_NAME"::varchar as C\_LAST\_NAME

,$1:"C\_PREFERRED\_CUST\_FLAG"::varchar as C\_PREFERRED\_CUST\_FLAG

,$1:"C\_BIRTH\_DAY"::decimal as C\_BIRTH\_DAY

,$1:"C\_BIRTH\_MONTH"::decimal as C\_BIRTH\_MONTH

,$1:"C\_BIRTH\_YEAR"::decimal as C\_BIRTH\_YEAR

,$1:"C\_BIRTH\_COUNTRY"::varchar as C\_BIRTH\_COUNTRY

,$1:"C\_LOGIN"::varchar as C\_LOGIN

,$1:"C\_EMAIL\_ADDRESS"::varchar as C\_EMAIL\_ADDRESS

,$1:"C\_LAST\_REVIEW\_DATE"::varchar as C\_LAST\_REVIEW\_DATE

FROM @cus\_stage(FILE\_FORMAT => 'iceberg\_format');

**CREATING ICEBERG TABLE FOR ITEM DATA**

CREATE or REPLACE iceberg  TABLE managed\_items(

I\_ITEM\_SK NUMBER,

I\_ITEM\_ID VARCHAR,

I\_REC\_START\_DATE DATE,

I\_REC\_END\_DATE DATE,

I\_ITEM\_DESC VARCHAR,

I\_CURRENT\_PRICE NUMBER,

I\_WHOLESALE\_COST NUMBER,

I\_BRAND\_ID NUMBER,

I\_BRAND VARCHAR,

I\_CLASS\_ID NUMBER,

I\_CLASS VARCHAR,

I\_CATEGORY\_ID NUMBER,

I\_CATEGORY VARCHAR,

I\_MANUFACT\_ID NUMBER,

I\_MANUFACT VARCHAR,

I\_SIZE VARCHAR,

I\_FORMULATION VARCHAR,

I\_COLOR VARCHAR,

I\_UNITS VARCHAR,

I\_CONTAINER VARCHAR,

I\_MANAGER\_ID NUMBER,

I\_PRODUCT\_NAME VARCHAR

)

CATALOG='SNOWFLAKE'

EXTERNAL\_VOLUME='snowlake\_managed\_exvol'

BASE\_LOCATION='managed-iceberg/'

as

select

$1:I\_ITEM\_SK:: NUMBER as I\_ITEM\_SK

,$1:I\_ITEM\_ID:: VARCHAR as I\_ITEM\_ID

,$1:I\_REC\_START\_DATE::DATE as I\_REC\_START\_DATE

,$1:I\_REC\_END\_DATE::DATE as I\_REC\_END\_DATE

,$1:I\_ITEM\_DESC::VARCHAR as I\_ITEM\_DESC

,$1:I\_CURRENT\_PRICE::NUMBER as I\_CURRENT\_PRICE

,$1:I\_WHOLESALE\_COST::NUMBER as I\_WHOLESALE\_COST

,$1:I\_BRAND\_ID::NUMBER as I\_BRAND\_ID

,$1:I\_BRAND::VARCHAR as I\_BRAND

,$1:I\_CLASS\_ID::NUMBER as I\_CLASS\_ID

,$1:I\_CLASS::VARCHAR as I\_CLASS

,$1:I\_CATEGORY\_ID::NUMBER as I\_CATEGORY\_ID

,$1:I\_CATEGORY::VARCHAR as I\_CATEGORY

,$1:I\_MANUFACT\_ID::NUMBER as I\_MANUFACT\_ID

,$1:I\_MANUFACT::VARCHAR as I\_MANUFACT

,$1:I\_SIZE::VARCHAR as I\_SIZE

,$1:I\_FORMULATION::VARCHAR as I\_FORMULATION

,$1:I\_COLOR::VARCHAR as I\_COLOR

,$1:I\_UNITS::VARCHAR as I\_UNITS

,$1:I\_CONTAINER::VARCHAR as I\_CONTAINER

,$1:I\_MANAGER\_ID::NUMBER as I\_MANAGER\_ID

,$1:I\_PRODUCT\_NAME::VARCHAR as I\_PRODUCT\_NAME

FROM @item\_stage(FILE\_FORMAT => 'iceberg\_format') ;

**To sync the metadata**

SELECT SYSTEM$GET\_ICEBERG\_TABLE\_INFORMATION('iceberg\_managed\_store\_returns');

**Incremental load into the iceberg table**

create or replace pipe managed\_snowpipe

auto\_ingest= true as

copy into iceberg\_managed\_store\_returns

from(select

$1:SR\_RETURNED\_DATE\_SK:: decimal as SR\_RETURNED\_DATE\_SK

,$1:SR\_RETURN\_TIME\_SK::decimal as SR\_RETURN\_TIME\_SK

,$1:SR\_ITEM\_SK::decimal as SR\_ITEM\_SK

,$1:SR\_CUSTOMER\_SK::decimal as SR\_CUSTOMER\_SK

,$1:SR\_CDEMO\_SK::decimal as SR\_CDEMO\_SK

,$1:SR\_HDEMO\_SK::decimal as SR\_HDEMO\_SK

,$1:SR\_ADDR\_SK::decimal as SR\_ADDR\_SK

,$1:SR\_STORE\_SK::decimal as SR\_STORE\_SK

,$1:SR\_REASON\_SK::decimal as SR\_REASON\_SK

,$1:SR\_TICKET\_NUMBER::decimal as SR\_TICKET\_NUMBER

,$1:SR\_RETURN\_QUANTITY::decimal as SR\_RETURN\_QUANTITY

,$1:SR\_RETURN\_AMT::decimal as SR\_RETURN\_AMT

,$1:SR\_RETURN\_TAX::decimal as SR\_RETURN\_TAX

,$1:SR\_RETURN\_AMT\_INC\_TAX::decimal as SR\_RETURN\_AMT\_INC\_TAX

,$1:SR\_FEE::decimal as SR\_FEE

,$1:SR\_RETURN\_SHIP\_COST::decimal as SR\_RETURN\_SHIP\_COST

,$1:SR\_REFUNDED\_CASH::decimal as SR\_REFUNDED\_CASH

,$1:SR\_REVERSED\_CHARGE::decimal as SR\_REVERSED\_CHARGE

,$1:SR\_STORE\_CREDIT::decimal as SR\_STORE\_CREDIT

,$1:SR\_NET\_LOSS::decimal as SR\_NET\_LOSS

from @external\_stage (file\_format => 'iceberg\_format'));

**Interoperability using Spark [Local system]**

spark-shell --packages org.apache.spark:spark-core\_2.12:3.3.2,org.apache.spark:spark-sql\_2.12:3.3.2,org.apache.iceberg:iceberg-spark3:0.13.1,org.apache.logging.log4j:log4j-core:2.20.0,org.apache.hadoop:hadoop-aws:3.3.2,com.amazonaws:aws-java-sdk-glue:1.12.471,com.amazonaws:aws-java-sdk-s3:1.12.544,net.snowflake:spark-snowflake\_2.12:2.12.0-spark\_3.4,net.snowflake:snowflake-jdbc:3.14.1,org.apache.hive:hive-metastore:3.1.3

import org.apache.spark.sql.\_

import org.apache.spark.SparkConf

val sparkConf = new SparkConf().setAppName("YourSparkApp").setMaster("local[\*]").set("spark.driver.memory", "2g")

val spark = SparkSession.builder().config(sparkConf) .master("local[3]").config("spark.sql.catalog.local", "org.apache.iceberg.spark.SparkCatalog").config("spark.sql.catalog.local.type", "hadoop").config("spark.hadoop.fs.s3a.endpoint", "s3.amazonaws.com").config("spark.hadoop.fs.s3a.region", "us-east-1").config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem").config("spark.hadoop.fs.s3a.access.key", "AKIA23KQLPGLSRCTCPOS").config("spark.hadoop.fs.s3a.secret.key", "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").appName("iceberg-spark").getOrCreate()

val snowflakeOptions = Map(

"sfURL" -> "https://anblicksorg\_aws.us-east-1.snowflakecomputing.com",

"sfUser" -> "SNOWLAKE",

"sfpassword" -> "SnowLake@202308",

"sfDatabase" -> "SNOWLAKE\_DB",           
"sfSchema" -> "SNOWFLAKE\_MANAGED",

"sfWarehouse" -> "SNOWLAKE\_WH",

"sfRole" -> "SNOWLAKE\_ARL"

)

val ff = spark.read.format("net.snowflake.spark.snowflake").options(snowflakeOptions).option("dbtable", "SNOWLAKE\_DB.SNOWFLAKE\_MANAGED.iceberg\_managed\_store\_returns").load("s3://snowlake-bucket/managed-iceberg/12325538234778022/").createOrReplaceTempView("iceberg\_returns");

spark.sql("SELECT count(\*) from iceberg\_returns").show(false)

spark.sql("SELECT SR\_RETURNED\_DATE\_SK,SR\_RETURN\_TIME\_SK,SR\_ITEM\_SK from iceberg\_returns where  SR\_RETURNED\_DATE\_SK IN ('24516134', '24516135') ").show(false)

**Schema Evolution**

---Rename the column in store returns:

ALTER iceberg table iceberg\_managed\_store\_returns RENAME column SR\_REFUNDED\_CASH to SR\_REFUNDED\_MONEY;

---Add the Column in store returns:

ALTER iceberg table iceberg\_managed\_store\_returns ADD column SR\_REPRESENTATIVE varchar;

----Drop the column in store returns:

ALTER iceberg table iceberg\_managed\_store\_returns DROP column SR\_REPRESENTATIVE;

---Insert the values in store returns:

INSERT INTO iceberg\_managed\_store\_returns (SR\_RETURNED\_DATE\_SK ,SR\_RETURN\_TIME\_SK,SR\_ITEM\_SK)

    VALUES ('24516134', '67000', '18888'),

           ('24516135', '70000', '19999');

---Update rows in store returns:

UPDATE iceberg\_managed\_store\_returns SET SR\_CUSTOMER\_SK = 3456778 WHERE SR\_RETURNED\_DATE\_SK=24516134;

UPDATE iceberg\_managed\_store\_returns SET SR\_CUSTOMER\_SK =123  WHERE SR\_RETURNED\_DATE\_SK=24516134;

---Delete rows in store returns:

DELETE FROM iceberg\_managed\_store\_returns WHERE SR\_RETURNED\_DATE\_SK IN ('24516134', '24516135');

**Data Masking Policy**

CREATE OR REPLACE MASKING POLICY customer\_mask AS (VAL STRING) RETURNS STRING ->

CASE

WHEN CURRENT\_ROLE() IN ('ACCOUNTADMIN') THEN VAL

ELSE '\*\*HIDDEN\*\*'

END;

ALTER ICEBERG TABLE if exists managed\_customer modify column C\_EMAIL\_ADDRESS SET masking policy customer\_mask;

**Row Access** **Policy** :

CREATE OR REPLACE ROW ACCESS POLICY item\_category

    AS (I\_CATEGORY string) returns boolean ->

    CASE   WHEN 'ACCOUNTADMIN' = CURRENT\_ROLE() then true

       WHEN 'SNOWLAKE\_ARL' = CURRENT\_ROLE() and I\_CATEGORY = 'Men' then true

    else false

end

ALTER iceberg table MANAGED\_ITEMS ADD row access policy item\_category on (I\_CATEGORY);

**Joins And Shares Using Secure Views** **In Managed Iceberg Tables**

--Creating Secure View on One Iceberg Table

CREATE OR REPLACE SECURE VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_MANAGED\_VIEW As

(select I\_ITEM\_SK,I\_ITEM\_ID,I\_ITEM\_DESC from snowlake\_db.snowFlake\_managed.MANAGED\_ITEMS);

--Creating Secure View on joining two Iceberg Tables

CREATE OR REPLACE SECURE VIEW sales\_info as

select \* from SNOWLAKE\_DB.SNOWFLAKE\_MANAGED.ICEBERG\_MANAGED\_STORE\_RETURNS st join SNOWLAKE\_DB.SNOWFLAKE\_MANAGED.MANAGED\_ITEMS item on item.I\_ITEM\_SK= st.SR\_ITEM\_SK;

--Creating Secure View on joining  Iceberg Table and Native Table

create or replace secure view iceberg\_native\_info as

select \* from SNOWLAKE\_DB.SNOWFLAKE\_MANAGED.native\_store\_returns nt join SNOWLAKE\_DB.SNOWFLAKE\_MANAGED.MANAGED\_ITEMS item on item.I\_ITEM\_SK= nt.SR\_ITEM\_SK; -- iceberg table and native table

--Creating Shares

create or replace share MANAGED\_SHARE;

GRANT USAGE ON DATABASE SNOWLAKE\_SHARE\_DB TO SHARE MANAGED\_SHARE;

GRANT USAGE ON SCHEMA SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE TO SHARE MANAGED\_SHARE;

GRANT REFERENCE\_USAGE on database SNOWLAKE\_DB to share MANAGED\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_MANAGED\_VIEW TO SHARE MANAGED\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.ICEBERG\_MANAGED\_VIEW TO SHARE MANAGED\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.sales\_info TO SHARE MANAGED\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.sales\_info TO SHARE MANAGED\_SHARE;

GRANT SELECT ON VIEW SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.iceberg\_native\_info TO SHARE MANAGED\_SHARE;

GRANT SELECT ON TABLE SNOWLAKE\_SHARE\_DB.SNOWLAKE\_SHARE.iceberg\_native\_info TO SHARE MANAGED\_SHARE;

ALTER SHARE MANAGED\_SHARE ADD ACCOUNT = jzb03580;