**Snowflake Metadata Share Guide**



|  |  |
| --- | --- |
| Product Version | 4.8.0 |
| Document Type | Snowshare Guide |
| Authors | Snowflake Data source Team |
| Reviewer | Red Team & Architects |
| Approver | CTO |
| Total Pages | 10 |
| Document Status | Draft |

Table Of Contents

* 1. Objectives 2
  2. Architecture 2
  3. Pre-requisite 3
  4. Share Snowflake Metadata with Snowflake secure share. 3
     1. Execute below statement to create the procedure and necessary functions. 4
     2. One time execution 14
     3. To continue execution on certain interval this is created as tasks. 14
  5. Share the Transient tables to unravel account. 15
  6. Receive the data in recipient account (sql mode). 16
  7. Share the Transient tables to unravel account (UI Mode). 16
  8. Receive the data in recipient account ui mode. 19
  9. Configure recipient account in unravel. 21

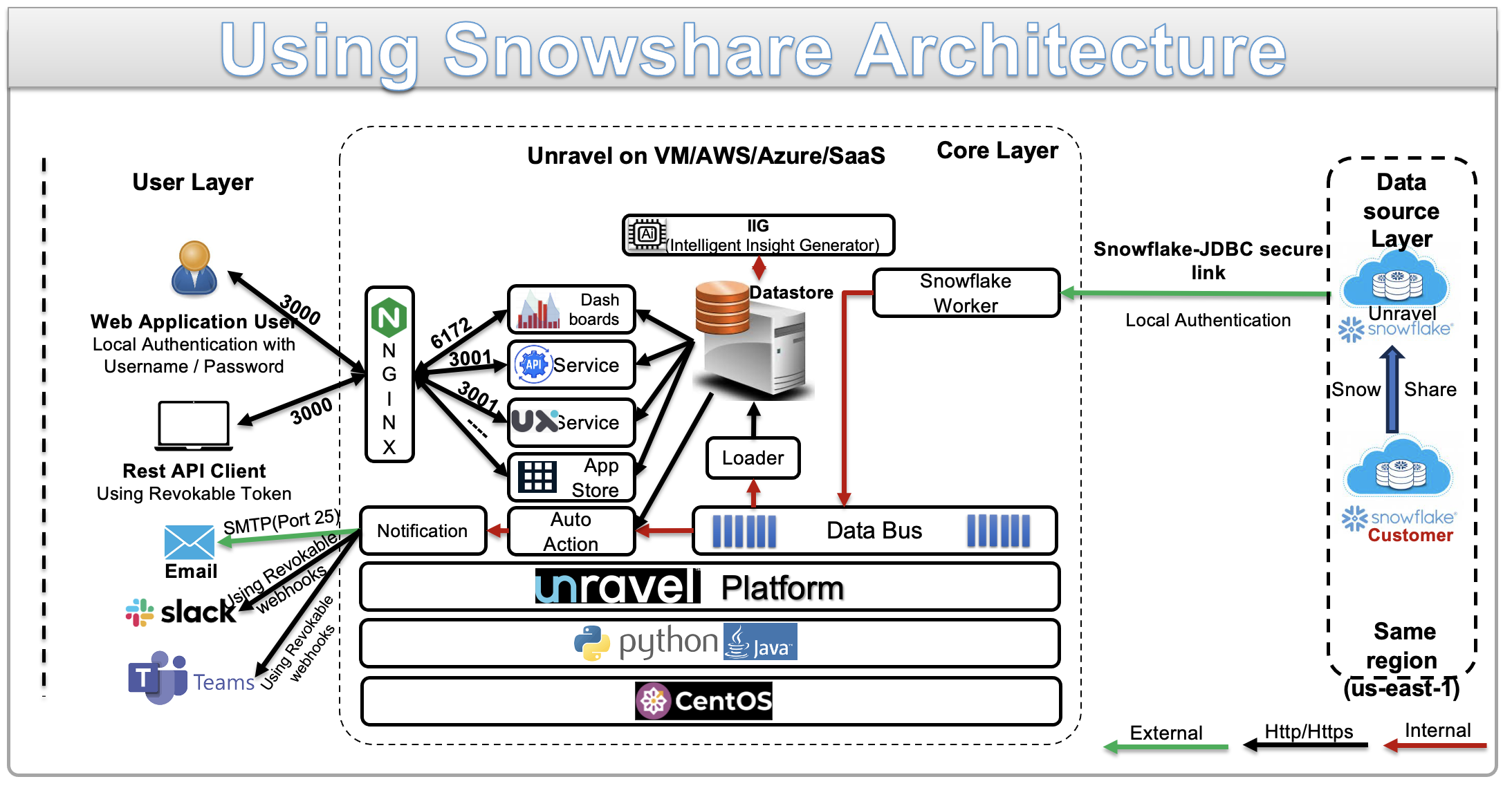
Document Version Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version # | Author | Remarks / Reason |
| 02-May-23 | 1.0 | Dev Team | New Document |

## Objectives

Health check download for snowflake unravel product.

## Architecture



## Pre-requisite

1. **Snowflake account Access through snowshare**
   * 1. Create a unravel user in snowflake
     2. Grant select for unravel user on schema SNOWFLAKE.ACCOUNT\_USAGE.
     3. Create a database unraveldb and a schema unravelschema to create a stored procedure required for metadata collection from account\_usage, information\_schema & Profile and store in local schema
     4. Execute procedure to create tables and serverless task to populate the metadata from the account metadata.
     5. Need to have Snowshare access to share the local schema to unravel snowflake account , Execute the snowshare script to give access to ”Unravel Snowflake account”.
     6. Based on the latency required the serverless/warehouse task will cost.

## Share Snowflake Metadata with Snowflake secure share.

## Execute below statement to create the procedure and necessary functions.

CREATE DATABASE IF NOT EXISTS UNRAVEL\_SHARE;

USE UNRAVEL\_SHARE;

CREATE SCHEMA IF NOT EXISTS SCHEMA\_4823\_T;

USE UNRAVEL\_SHARE.SCHEMA\_4823\_T;

CREATE OR REPLACE PROCEDURE CREATE\_TABLES(DB STRING, SCHEMA STRING)

RETURNS STRING NOT NULL

LANGUAGE SQL

EXECUTE AS CALLER

AS

DECLARE

use\_statement VARCHAR;

res RESULTSET;

BEGIN

use\_statement := 'USE ' || DB || '.' || SCHEMA;

res := (EXECUTE IMMEDIATE :use\_statement);

CREATE OR REPLACE TABLE replication\_log (

eventDate DATE DEFAULT current\_date,

executionStatus VARCHAR(1000) DEFAULT NULL,

remarks VARCHAR(1000)

);

CREATE OR REPLACE TRANSIENT TABLE WAREHOUSE\_METERING\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_METERING\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE WAREHOUSE\_EVENTS\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_EVENTS\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE WAREHOUSE\_LOAD\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_LOAD\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE TABLES WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.TABLES;

CREATE OR REPLACE TRANSIENT TABLE METERING\_DAILY\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.METERING\_DAILY\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE METERING\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.METERING\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE DATABASE\_REPLICATION\_USAGE\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.DATABASE\_REPLICATION\_USAGE\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE REPLICATION\_GROUP\_USAGE\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.REPLICATION\_GROUP\_USAGE\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE DATABASE\_STORAGE\_USAGE\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.DATABASE\_STORAGE\_USAGE\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE STAGE\_STORAGE\_USAGE\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.STAGE\_STORAGE\_USAGE\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE SEARCH\_OPTIMIZATION\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.SEARCH\_OPTIMIZATION\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE DATA\_TRANSFER\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.DATA\_TRANSFER\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE AUTOMATIC\_CLUSTERING\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.AUTOMATIC\_CLUSTERING\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE TAG\_REFERENCES WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.TAG\_REFERENCES;

CREATE OR REPLACE TRANSIENT TABLE QUERY\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE ACCESS\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 LIKE SNOWFLAKE.ACCOUNT\_USAGE.ACCESS\_HISTORY;

CREATE OR REPLACE TRANSIENT TABLE IS\_QUERY\_HISTORY WITH DATA\_RETENTION\_TIME\_IN\_DAYS=0 AS SELECT \* FROM TABLE(INFORMATION\_SCHEMA.QUERY\_HISTORY()) WHERE 1=0;

RETURN 'SUCCESS';

END;

-- PROCEDURE FOR REPLICATE ACCOUNT\_USAGE

CREATE OR REPLACE PROCEDURE REPLICATE\_ACCOUNT\_USAGE(DB STRING, SCHEMA STRING, LOOK\_BACK\_DAYS INTEGER)

RETURNS STRING NOT NULL

LANGUAGE SQL

EXECUTE AS CALLER

AS

DECLARE

use\_statement VARCHAR;

res RESULTSET;

BEGIN

use\_statement := 'USE ' || DB || '.' || SCHEMA;

res := (EXECUTE IMMEDIATE :use\_statement);

TRUNCATE TABLE IF EXISTS WAREHOUSE\_METERING\_HISTORY;

INSERT INTO WAREHOUSE\_METERING\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_METERING\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date) ;

TRUNCATE TABLE IF EXISTS WAREHOUSE\_EVENTS\_HISTORY ;

INSERT INTO WAREHOUSE\_EVENTS\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_EVENTS\_HISTORY HIS WHERE HIS.TIMESTAMP > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date) ;

TRUNCATE TABLE IF EXISTS WAREHOUSE\_LOAD\_HISTORY ;

INSERT INTO WAREHOUSE\_LOAD\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_LOAD\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS TABLES ;

INSERT INTO TABLES SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.TABLES;

TRUNCATE TABLE IF EXISTS METERING\_DAILY\_HISTORY ;

INSERT INTO METERING\_DAILY\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.METERING\_DAILY\_HISTORY HIS WHERE HIS.USAGE\_DATE > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS METERING\_HISTORY ;

INSERT INTO METERING\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.METERING\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS DATABASE\_REPLICATION\_USAGE\_HISTORY ;

INSERT INTO DATABASE\_REPLICATION\_USAGE\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.DATABASE\_REPLICATION\_USAGE\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS REPLICATION\_GROUP\_USAGE\_HISTORY ;

INSERT INTO REPLICATION\_GROUP\_USAGE\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.REPLICATION\_GROUP\_USAGE\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS DATABASE\_STORAGE\_USAGE\_HISTORY ;

INSERT INTO DATABASE\_STORAGE\_USAGE\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.DATABASE\_STORAGE\_USAGE\_HISTORY HIS WHERE HIS.USAGE\_DATE > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS STAGE\_STORAGE\_USAGE\_HISTORY ;

INSERT INTO STAGE\_STORAGE\_USAGE\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.STAGE\_STORAGE\_USAGE\_HISTORY HIS WHERE HIS.USAGE\_DATE > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS SEARCH\_OPTIMIZATION\_HISTORY ;

INSERT INTO SEARCH\_OPTIMIZATION\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.SEARCH\_OPTIMIZATION\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS DATA\_TRANSFER\_HISTORY ;

INSERT INTO DATA\_TRANSFER\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.DATA\_TRANSFER\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS AUTOMATIC\_CLUSTERING\_HISTORY ;

INSERT INTO AUTOMATIC\_CLUSTERING\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.AUTOMATIC\_CLUSTERING\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY ;

INSERT INTO SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS TAG\_REFERENCES ;

INSERT INTO TAG\_REFERENCES SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.TAG\_REFERENCES ;

TRUNCATE TABLE IF EXISTS QUERY\_HISTORY ;

INSERT INTO QUERY\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY HIS WHERE HIS.START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

TRUNCATE TABLE IF EXISTS ACCESS\_HISTORY ;

INSERT INTO ACCESS\_HISTORY SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.ACCESS\_HISTORY HIS WHERE HIS.QUERY\_START\_TIME > DATEADD(Day ,-:LOOK\_BACK\_DAYS, current\_date);

RETURN 'SUCCESS';

END;

--PROCEDURE FOR REPLICATE REALTIME QUERY

CREATE OR REPLACE PROCEDURE REPLICATE\_REALTIME\_QUERY(DB STRING, SCHEMA STRING, LOOK\_BACK\_HOURS INTEGER)

RETURNS STRING NOT NULL

LANGUAGE SQL

EXECUTE AS CALLER

AS

DECLARE

use\_statement VARCHAR;

res RESULTSET;

BEGIN

use\_statement := 'USE ' || DB || '.' || SCHEMA;

res := (EXECUTE IMMEDIATE :use\_statement);

TRUNCATE TABLE IF EXISTS IS\_QUERY\_HISTORY ;

INSERT INTO IS\_QUERY\_HISTORY SELECT \* FROM TABLE(INFORMATION\_SCHEMA.QUERY\_HISTORY(dateadd('hours',-:LOOK\_BACK\_HOURS ,current\_timestamp()),current\_timestamp(),10000)) order by start\_time ;

RETURN 'SUCCESS';

END;

CCREATE OR REPLACE PROCEDURE create\_query\_profile(dbname string, schemaname string, cost string, days String)

returns VARCHAR(25200)

LANGUAGE javascript

AS

$$

var create\_query\_profile\_task = "create\_query\_profile ---> Getting Query Profile data and inserting into Query\_profile table";

var task="profile\_task";

function logError(err, taskName)

{

var fail\_sql = "INSERT INTO REPLICATION\_LOG VALUES (current\_timestamp,'FAILED', "+"'"+ err +"'"+", "+"'"+ taskName +"'"+");" ;

sql\_command1 = snowflake.createStatement({sqlText: fail\_sql} );

sql\_command1.execute();

}

function insertToReplicationLog(status, message, taskName)

{

var query\_profile\_status = "INSERT INTO REPLICATION\_LOG VALUES (current\_timestamp, "+"'"+status +"'"+", "+"'"+ message +"'"+", "+"'"+ taskName +"'"+");" ;

sql\_command1 = snowflake.createStatement({sqlText: query\_profile\_status} );

sql\_command1.execute();

}

try

{

var query = 'CREATE DATABASE IF NOT EXISTS ' + DBNAME + ';';

var stmt = snowflake.createStatement({sqlText:query})

stmt.execute();

result = "Database: " + DBNAME + " creation is success";

}

catch (err)

{

logError(err, create\_query\_profile\_task)

return "Failed to create DB " + DBNAME + ", error: " + err;

}

try

{

var query = 'CREATE SCHEMA IF NOT EXISTS ' + DBNAME + '.' + SCHEMANAME + ';';

var stmt = snowflake.createStatement({sqlText:query})

stmt.execute();

result += "\nSchema: " + SCHEMANAME + " creation is success";

}

catch (err)

{ logError(err, create\_query\_profile\_task)

return "Failed to create the schema "+ SCHEMANAME + ", error: " + err;

}

var schemaName = SCHEMANAME;

var dbName = DBNAME;

var cost = parseFloat(COST);

var lookBackDays = -parseInt(DAYS);

const queries = [];

queries[0] = 'CREATE TRANSIENT TABLE IF NOT EXISTS ' + dbName + '.' + schemaName + '.QUERY\_PROFILE (QUERY\_ID VARCHAR(16777216),STEP\_ID NUMBER(38, 0),OPERATOR\_ID NUMBER(38,0),PARENT\_OPERATORS ARRAY, OPERATOR\_TYPE VARCHAR(16777216),OPERATOR\_STATISTICS VARIANT,EXECUTION\_TIME\_BREAKDOWN VARIANT, OPERATOR\_ATTRIBUTES VARIANT);';

queries[1] = "CREATE OR REPLACE TEMPORARY TABLE "+ dbName + "." + schemaName + ".query\_history\_temp AS SELECT query\_id, unit \* execution\_time \* query\_load\_percent / 100 / (3600 \* 1000) as cost from( SELECT query\_id, query\_load\_percent, CASE WHEN WAREHOUSE\_SIZE = 'X-Small' THEN 1 WHEN WAREHOUSE\_SIZE = 'Small' THEN 2 WHEN WAREHOUSE\_SIZE = 'Medium' THEN 4 WHEN WAREHOUSE\_SIZE = 'Large' THEN 6 WHEN WAREHOUSE\_SIZE = 'X-Large' THEN 8 WHEN WAREHOUSE\_SIZE = '2X-Large' THEN 10 WHEN WAREHOUSE\_SIZE = '3X-Large' THEN 12 WHEN WAREHOUSE\_SIZE = '4X-Large' THEN 14 WHEN WAREHOUSE\_SIZE = '5X-Large' THEN 16 WHEN WAREHOUSE\_SIZE = '6X-Large' THEN 18 ELSE 1 END as unit, execution\_time FROM SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY WHERE START\_TIME > dateadd(day, "+ lookBackDays +", current\_date) ORDER BY start\_time) where cost is not null AND cost > " +cost+";";

queries[2] = "SELECT count(1) FROM "+ dbName + "." + schemaName + ".query\_history\_temp";

var returnVal = "SUCCESS";

var error = "";

var total\_query\_count = 0;

var failed\_query\_count = 0;

for (let i = 0; i < queries.length; i++) {

var stmt = snowflake.createStatement({sqlText:queries[i]});

try

{

var res = stmt.execute();

if(i==2)

{

res.next();

total\_query\_count = res.getColumnValue(1)

var message ="Total records = "+ total\_query\_count;

insertToReplicationLog("started",message,task);

}

}

catch (err)

{

logError(err, create\_query\_profile\_task)

error += "Failed: " + err;

}

}

if(error.length > 0 ) {

return error;

}

var actualQueryId = 'SELECT tmp.query\_id FROM '+ dbName + '.' + schemaName + '.query\_history\_temp tmp WHERE NOT EXISTS (SELECT query\_id FROM QUERY\_PROFILE WHERE query\_id = tmp.query\_id);';

var profileInsert = 'INSERT INTO ' + dbName + '.' + schemaName + '.QUERY\_PROFILE select \* from table(get\_query\_operator\_stats(?));';

var stmt = snowflake.createStatement({sqlText: actualQueryId});

var query\_count = 0;

try

{

var result\_set1 = stmt.execute();

while (result\_set1.next()) {

var queryId = result\_set1.getColumnValue(1);

var profileInsertStmt = snowflake.createStatement({sqlText: profileInsert, binds:[queryId]});

profileInsertStmt.execute();

query\_count++;

if (query\_count % 100 == 0){

var message ="Total records = "+ total\_query\_count +", completed = "+query\_count+", failed = "+failed\_query\_count;

insertToReplicationLog("running", message, task);

}

}

}

catch (err)

{

logError(err, create\_query\_profile\_task)

error += "Failed: " + err;

}

var message ="Total records = "+ total\_query\_count +", completed = "+query\_count+", failed = "+failed\_query\_count;

insertToReplicationLog("completed", message, task);

return returnVal;

$$;

CREATE OR REPLACE PROCEDURE warehouse\_proc(dbname STRING, schemaname STRING)

RETURNS VARCHAR(252)

LANGUAGE JAVASCRIPT

EXECUTE AS CALLER

AS

$$

var warehouse\_proc\_task = "warehouse\_proc ---> Warehouses and Warehouse\_Parameter Table Creation";

function logError(err, taskName)

{

var fail\_sql = "INSERT INTO REPLICATION\_LOG VALUES (current\_timestamp,'FAILED', "+"'"+ err +"'"+", "+"'"+ taskName +"'"+");" ;

sql\_command1 = snowflake.createStatement({sqlText: fail\_sql} );

sql\_command1.execute();

}

try {

var query = 'CREATE DATABASE IF NOT EXISTS ' + DBNAME + ';';

var stmt = snowflake.createStatement({

sqlText: query

})

stmt.execute();

result = "Database: " + DBNAME + " creation is success";

} catch (err) {

logError(err, warehouse\_proc\_task);

return "Failed to create DB " + DBNAME + ", error: " + err;

}

try {

var query = 'CREATE SCHEMA IF NOT EXISTS ' + DBNAME + '.' + SCHEMANAME + ';';

var stmt = snowflake.createStatement({

sqlText: query

})

stmt.execute();

result += "\nSchema: " + SCHEMANAME + " creation is success";

} catch (err) {

logError(err, warehouse\_proc\_task);

return "Failed to create the schema " + SCHEMANAME + ", error: " + err;

}

var returnVal = "SUCCESS";

var error = "";

try {

// 1. create warehouse table if not exist

var createWarehouseTable = 'CREATE TRANSIENT TABLE IF NOT EXISTS ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSES(NAME VARCHAR(16777216), STATE VARCHAR(16777216), TYPE VARCHAR(16777216), SIZE VARCHAR(16777216), MIN\_CLUSTER\_COUNT NUMBER(38,0), MAX\_CLUSTER\_COUNT NUMBER(38,0), STARTED\_CLUSTERS NUMBER(38,0), RUNNING NUMBER(38,0), QUEUED NUMBER(38,0), IS\_DEFAULT VARCHAR(1), IS\_CURRENT VARCHAR(1), AUTO\_SUSPEND NUMBER(38,0), AUTO\_RESUME VARCHAR(16777216), AVAILABLE VARCHAR(16777216), PROVISIONING VARCHAR(16777216), QUIESCING VARCHAR(16777216), OTHER VARCHAR(16777216), CREATED\_ON TIMESTAMP\_LTZ(9), RESUMED\_ON TIMESTAMP\_LTZ(9), UPDATED\_ON TIMESTAMP\_LTZ(9), OWNER VARCHAR(16777216), COMMENT VARCHAR(16777216), ENABLE\_QUERY\_ACCELERATION VARCHAR(16777216), QUERY\_ACCELERATION\_MAX\_SCALE\_FACTOR NUMBER(38,0), RESOURCE\_MONITOR VARCHAR(16777216), ACTIVES NUMBER(38,0), PENDINGS NUMBER(38,0), FAILED NUMBER(38,0), SUSPENDED NUMBER(38,0), UUID VARCHAR(16777216), SCALING\_POLICY VARCHAR(16777216), BUDGET VARCHAR(16777216));';

var createWarehouseTableStmt = snowflake.createStatement({

sqlText: createWarehouseTable

});

createWarehouseTableStmt.execute();

// 2. truncate table

var truncateWarehouse = 'TRUNCATE TABLE IF EXISTS ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSES;';

var truncateWarehouseStmt = snowflake.createStatement({

sqlText: truncateWarehouse

});

truncateWarehouseStmt.execute();

// 3. run show warehouses

var showWarehouse = 'SHOW WAREHOUSES;';

var showWarehouseStmt = snowflake.createStatement({

sqlText: showWarehouse

});

var resultSet = showWarehouseStmt.execute();

// 4. insert to warehouse

var insertToWarehouse = 'INSERT INTO ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSES SELECT \* FROM TABLE(result\_scan(last\_query\_id()));';

var insertToWarehouseStmt = snowflake.createStatement({

sqlText: insertToWarehouse

});

insertToWarehouseStmt.execute();

} catch (err) {

logError(err, warehouse\_proc\_task);

error += "Failed: " + err;

}

try {

//1. create warehouse parameters table

var createWP = 'CREATE TRANSIENT TABLE IF NOT EXISTS ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSE\_PARAMETERS (WAREHOUSE VARCHAR(1000), KEY VARCHAR(1000), VALUE VARCHAR(1000), DEFUALT VARCHAR(1000),LEVEL VARCHAR(1000), DESCRIPTION VARCHAR(10000),TYPE VARCHAR(100));';

var createWPStmt = snowflake.createStatement({

sqlText: createWP

});

createWPStmt.execute();

//2. trunate warehouse parameter tables

var truncateWarehouseParameter = 'TRUNCATE TABLE IF EXISTS ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSE\_PARAMETERS;';

var truncateWarehouseParameterStmt = snowflake.createStatement({

sqlText: truncateWarehouseParameter

});

truncateWarehouseParameterStmt.execute();

} catch (err) {

logError(err, warehouse\_proc\_task);

error += "Failed: " + err;

}

try {

//3.Get warehouse details

var wn = 'SELECT \* FROM ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSES;';

var wnStmt = snowflake.createStatement({

sqlText: wn

});

var resultSet1 = wnStmt.execute();

while (resultSet1.next()) {

var whName = resultSet1.getColumnValue(1);

//4. show warehouse parameters

var showWP = 'SHOW PARAMETERS IN WAREHOUSE ' + whName + ';';

var showWPStmt = snowflake.createStatement({

sqlText: showWP

});

showWPStmt.execute();

//5. insert into WAREHOUSE\_PARAMETERS table

var wpInsert = 'INSERT INTO ' + DBNAME + '.' + SCHEMANAME + '.WAREHOUSE\_PARAMETERS SELECT ' + "'" + whName + "'" + ',\* FROM TABLE (result\_scan(last\_query\_id()));';

var wpInsertStmt = snowflake.createStatement({

sqlText: wpInsert

});

wpInsertStmt.execute();

}

} catch (err) {

error += "Failed: " + err;

return logError(err, warehouse\_proc\_task);

}

if (error.length > 0) {

return error;

}

return returnVal;

$$;

## One time execution

CALL CREATE\_TABLES('UNRAVEL\_SHARE','SCHEMA\_4823\_T');

CALL REPLICATE\_ACCOUNT\_USAGE('UNRAVEL\_SHARE','SCHEMA\_4823\_T',2);

CALL WAREHOUSE\_PROC('UNRAVEL\_SHARE','SCHEMA\_4823\_T');

CALL REPLICATE\_REALTIME\_QUERY('UNRAVEL\_SHARE','SCHEMA\_4823\_T',10);

CALL CREATE\_QUERY\_PROFILE(dbname => 'UNRAVEL\_SHARE',schemaname => 'SCHEMA\_4823\_T', cost => '1', days => '1');

## To continue execution on certain interval this is created as tasks.

-- create account usage tables Task

CREATE OR REPLACE TASK replicate\_metadata

WAREHOUSE = UNRAVELDATA

SCHEDULE = '60 MINUTE'

AS

CALL REPLICATE\_ACCOUNT\_USAGE('UNRAVEL\_SHARE','SCHEMA\_4823\_T',2);

-- create warehouse replicate Task

CREATE OR REPLACE TASK createWarehouseTable

WAREHOUSE = UNRAVELDATA

SCHEDULE = '60 MINUTE'

AS

CALL warehouse\_proc('UNRAVEL\_SHARE','SCHEMA\_4823\_T');

-- create profile replicate task

CREATE OR REPLACE TASK createProfileTable

WAREHOUSE = UNRAVELDATA

SCHEDULE = '60 MINUTE'

AS

CALL create\_query\_profile(dbname => 'UNRAVEL\_SHARE',schemaname => 'SCHEMA\_4823\_T', cost => '1', days => '1');

-- create Task for replicating information schema query history

CREATE OR REPLACE TASK replicate\_realtime\_query

WAREHOUSE = UNRAVELDATA

SCHEDULE = '30 MINUTE'

AS

CALL REPLICATE\_REALTIME\_QUERY('UNRAVEL\_SHARE','SCHEMA\_4823\_T',10);

-- START ALL THE TASKS

ALTER TASK replicate\_metadata RESUME;

ALTER TASK createWarehouseTable RESUME;

ALTER TASK createProfileTable RESUME;

ALTER TASK replicate\_realtime\_query RESUME;

## Share the Transient tables to unravel account.

From SQL (Recommended):

-- Share tables

Create share RICOH\_UNRAVEL\_SHARE;

Grant Usage on database UNRAVEL\_SHARE to share RICOH\_UNRAVEL\_SHARE;

Grant Usage on schema SCHEMA\_4823\_T to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE WAREHOUSE\_METERING\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE WAREHOUSE\_EVENTS\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE WAREHOUSE\_LOAD\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE TABLES to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE METERING\_DAILY\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE METERING\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE DATABASE\_REPLICATION\_USAGE\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE REPLICATION\_GROUP\_USAGE\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE DATABASE\_STORAGE\_USAGE\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE STAGE\_STORAGE\_USAGE\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE SEARCH\_OPTIMIZATION\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE DATA\_TRANSFER\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE AUTOMATIC\_CLUSTERING\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE SNOWPIPE\_STREAMING\_FILE\_MIGRATION\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE TAG\_REFERENCES to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE QUERY\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE ACCESS\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE IS\_QUERY\_HISTORY to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE WAREHOUSE\_PARAMETERS to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE WAREHOUSES to share RICOH\_UNRAVEL\_SHARE;

GRANT SELECT ON TABLE QUERY\_PROFILE to share RICOH\_UNRAVEL\_SHARE;

alter share RICOH\_UNRAVEL\_SHARE add accounts = HFB47355;

## Receive the data in recipient account (sql mode).

--Validate that the inbound share is available to the consumer account.

Show shares like '%UNRAVEL%';

Use role Accountadmin;

create database RICOH\_SHARE from share FWTTICE.PRIMARY\_PG.RICOH\_UNRAVEL\_SHARE;

## Share the Transient tables to unravel account (UI Mode).

Sharing through UI :

Eg url : <https://app.snowflake.com/fwttice/primary_pg/#/data/shared/outbound>

Replace your account in above url intead of “fwttice”

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now the tables are shared to **HFB47355 Account**

## Receive the data in recipient account ui mode.

Login as account admin.

A screenshot of a computer

Description automatically generated

Click here.

Select the role

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now data is available in recipient account

A screenshot of a computer

Description automatically generated

## Configure recipient account in unravel.

Show shares like '%UNRAVEL%';

Use role Accountadmin;

create database RICOH\_SHARE from share FWTTICE.PRIMARY\_PG.RICOH\_UNRAVEL\_SHARE;

Database : RICOH\_SHARE

Other configurations are same.