**External Stage on S3**

1. **Create User in AWS with Programmatic access and copy the credentials**

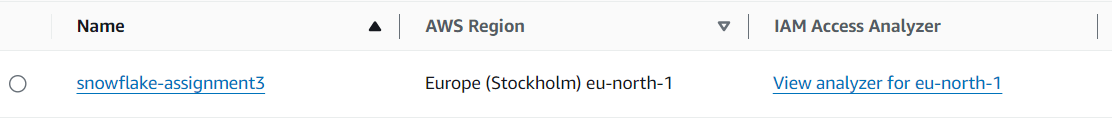
A screenshot of a computer

Description automatically generated

**user**



**b** **. Create s3 bucket**



**-------------------------- Create a file format for JSON------------------------------------------------**

CREATE OR REPLACE FILE FORMAT MY\_JSON\_FORMAT

TYPE = 'JSON'

STRIP\_OUTER\_ARRAY = TRUE

;

**c. Create Stage: Use below SQL statement in Snowflake to create external stage on s3(AWS).**

CREATE OR REPLACE STAGE ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_DATA\_JSON\_A3

URL='s3://snowflake-assignment3/snowflake-assignment3/'

CREDENTIALS = (AWS\_KEY\_ID = 'AKIA5FTZBH4PN46D3RUZ'

AWS\_SECRET\_KEY = '7DvFHclzbrkuvrCZhC0pienObfqUPTg2Lk1SxvXz')

FILE\_FORMAT = ASSIGNMENT3.PERSON\_SCHEMA3.MY\_JSON\_FORMAT;

**-------------------------------------------------Checking--------------------------------------------**

list @ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_DATA\_JSON\_A3;

**---------------------------------------------------Staging-------------------------------------------------**

desc stage ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_DATA\_JSON\_A3;

**d. CREATE table in Snowflake with VARIANT column.**

CREATE OR REPLACE TABLE PERSON (PERSONDATA VARIANT);

**e. Create a Snowpipe with Auto Ingest Enable**

CREATE OR REPLACE PIPE ASSIGNMENT3.PERSON\_SCHEMA3.person\_pipe AUTO\_INGEST = TRUE AS

COPY INTO PERSON

FROM (

SELECT $1

FROM @ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_DATA\_JSON\_A3

)

FILE\_FORMAT = (FORMAT\_NAME = 'ASSIGNMENT3.PERSON\_SCHEMA3.MY\_JSON\_FORMAT')

ON\_ERROR = 'CONTINUE';

**-- lISTING**

SHOW PIPES;

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

-- Below are few ways we can validation if Snowpipe rUn

-- successfully.

**-- 1 . Check the pipe status using below command, it shows**

**RUNNIG and it also shows pendingFileCount.**

select system$pipe\_status('person\_pipe');

**Check COPY\_HISTORY for the table you are loading data to. If**

**there is any error with Data Load, you can find that error here to**

**debug the Load issue**

select \* from table (information\_schema.copy\_history(table\_name=>'person',start\_time=>dateadd(hours,-1, current\_timestamp())));

**3. Finally check if data is loaded to table by querying the table**

SELECT \* FROM PERSON;

create or replace stream person\_stream on table person;

desc stream person\_stream;

**----------Checking stream data----------------**

SELECT \* FROM person\_stream;

**-- to see the arn on notification**

desc pipe PERSON\_PIPE;

**-- Table for storing person\_age information**

CREATE TABLE PERSON\_AGE (

Name VARCHAR(100),

Age INT

);

**-- Table for storing person\_location information**

CREATE TABLE PERSON\_LOCATION (

Name VARCHAR(100),

Location VARCHAR(100)

);

**2. Create a table to Load the unnested data from PERSON\_NESTED.**

CREATE OR REPLACE TABLE PERSON\_MASTER (

Id int,

Name STRING,

Age INT,

Location STRING,

Zip STRING

);

**Change Data Capture using Streams, Tasks and Merge.**

**1.Create Streams on PERSON\_NESTED table to capture the change data on PERSON\_NESTED table and use TASKS to Run SQL/Stored Procedure to Unnested the data from PERSON\_NESTED and create PERSON\_MASTER table.**

CREATE OR REPLACE PROCEDURE ASSIGNMENT3.PERSON\_SCHEMA3.upload\_person\_data()

RETURNS STRING

LANGUAGE JAVASCRIPT

EXECUTE AS OWNER

AS

$$

var sqlTextMerge = `

MERGE INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER P

USING (

SELECT

S.PERSONDATA:ID::INT AS ID,

S.PERSONDATA:Name::STRING AS NAME,

S.PERSONDATA:age::STRING AS AGE,

S.PERSONDATA:location::STRING AS LOCATION,

S.PERSONDATA:zip::STRING AS ZIP

FROM ASSIGNMENT3.PERSON\_SCHEMA3.person\_stream S

WHERE S.METADATA$ACTION = 'INSERT' OR S.METADATA$ISUPDATE = 'TRUE'

) S

ON P.ID = S.ID

WHEN MATCHED THEN

UPDATE SET

P.NAME = S.NAME,

P.AGE = S.AGE,

P.LOCATION = S.LOCATION,

P.ZIP = S.ZIP

WHEN NOT MATCHED THEN

INSERT (ID, NAME, AGE, LOCATION, ZIP)

VALUES (S.ID, S.NAME, S.AGE, S.LOCATION, S.ZIP);

`;

var sqlTextInsert1 = `

INSERT INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_AGE (Name, Age)

SELECT NAME, AGE

FROM ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER;

`;

var sqlTextInsert2 = `

INSERT INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_LOCATION (Name, LOCATION)

SELECT NAME, LOCATION

FROM ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER

WHERE LOCATION IS NOT NULL;

`;

// Execute the MERGE operation

snowflake.execute({sqlText: sqlTextMerge});

// Execute the first INSERT statement

snowflake.execute({sqlText: sqlTextInsert1});

// Execute the second INSERT statement

snowflake.execute({sqlText: sqlTextInsert2});

return `SUCCESS`;

$$;

**3. Create a TASK which run every 1 min and look for data in Stream PERSON\_NESTED\_STREAM, if data found in Stream then task will EXECUTE if not TASK will be SKIPPED without any doing anything.**

CREATE OR REPLACE TASK ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_TASK

WAREHOUSE='COMPUTE\_WH'

SCHEDULE='1 minute'

WHEN SYSTEM$STREAM\_HAS\_DATA('ASSIGNMENT3.PERSON\_SCHEMA3.person\_stream') AS

CALL ASSIGNMENT3.PERSON\_SCHEMA3.upload\_person\_data()

;

//refresh the task

alter pipe person\_pipe refresh;

//resume the task

ALTER TASK PERSON\_TASK RESUME;

**-- CHECKIng THE DATA using query.**

**//checking person\_nested data**

select \* from person;

**//checking person\_ master data**

SELECT \* FROM PERSON\_MASTER;

**//checking person\_ age data**

SELECT \* FROM PERSON\_AGE;

**//checking person\_ location data**

SELECT \* FROM PERSON\_LOCATION;

ELT IN SNOWFLAKE USING STORED PROCEDURE

a) Create stored procedure to run Multiple SQL statements to automate data Load from PERSON\_MASTER to two tables PERSON\_AGE(Name, Age) and PERSON\_LOCATION(Name, Location). This stored procedure should be called by TASK.

b) Stored Procedure Call :

c) CALL PERSON\_MASTER\_PROCEDURE(arguments1); Create Stored Procedure which runs below 2 SQLs.

1.Insert data into Location table from Person Master table.

2. Insert data into Age table from Person Master table

CREATE OR REPLACE PROCEDURE ASSIGNMENT3.PERSON\_SCHEMA3.upload\_person\_data()

RETURNS STRING

LANGUAGE JAVASCRIPT

EXECUTE AS OWNER

AS

$$

var sqlTextMerge = `

MERGE INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER P

USING (

SELECT

S.PERSONDATA:ID::INT AS ID,

S.PERSONDATA:Name::STRING AS NAME,

S.PERSONDATA:age::STRING AS AGE,

S.PERSONDATA:location::STRING AS LOCATION,

S.PERSONDATA:zip::STRING AS ZIP

FROM ASSIGNMENT3.PERSON\_SCHEMA3.person\_stream S

WHERE S.METADATA$ACTION = 'INSERT' OR S.METADATA$ISUPDATE = 'TRUE'

) S

ON P.ID = S.ID

WHEN MATCHED THEN

UPDATE SET

P.NAME = S.NAME,

P.AGE = S.AGE,

P.LOCATION = S.LOCATION,

P.ZIP = S.ZIP

WHEN NOT MATCHED THEN

INSERT (ID, NAME, AGE, LOCATION, ZIP)

VALUES (S.ID, S.NAME, S.AGE, S.LOCATION, S.ZIP);

`;

var sqlTextInsert1 = `

INSERT INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_AGE (Name, Age)

SELECT NAME, AGE

FROM ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER;

`;

var sqlTextInsert2 = `

INSERT INTO ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_LOCATION (Name, LOCATION)

SELECT NAME, LOCATION

FROM ASSIGNMENT3.PERSON\_SCHEMA3.PERSON\_MASTER

WHERE LOCATION IS NOT NULL;

`;

// Execute the MERGE operation

snowflake.execute({sqlText: sqlTextMerge});

// Execute the first INSERT statement

snowflake.execute({sqlText: sqlTextInsert1});

// Execute the second INSERT statement

snowflake.execute({sqlText: sqlTextInsert2});

return `SUCCESS`;

$$;

**truncate data from all table**

TRUNCATE TABLE PERSON;

TRUNCATE TABLE PERSON\_MASTER;

TRUNCATE TABLE PERSON\_LOCATION;

TRUNCATE TABLE person\_age;