**What Is a Stream?**

* A stream object records DML changes made to tables including inserts,updates, deletes.
* Also, Stream stores metadata about each change, so that actions can be taken using this metadata.
* We call this process as change data capture (CDC)
* Streams tracks all row level changes to a source table using offset but doesn’t store the changed data.
* Once these changes are consumed by the Target table, this offset moves to the next point.
* Streams can be combined with tasks to set continuous data pipelines.
* **Snow pipe + Stream + Task  Continuous Data Load.**

**Metadata of Streams:**

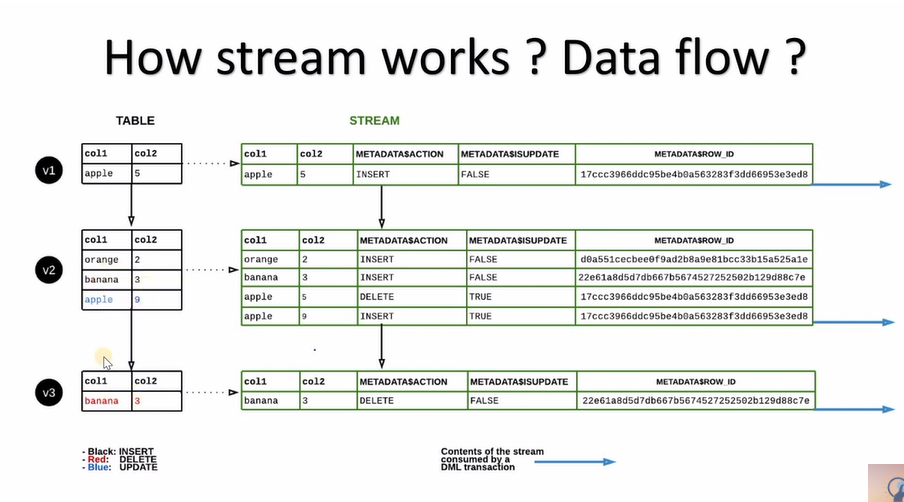
**Along with the changes made to the source table, Streams maintain 3 metadata fields.**

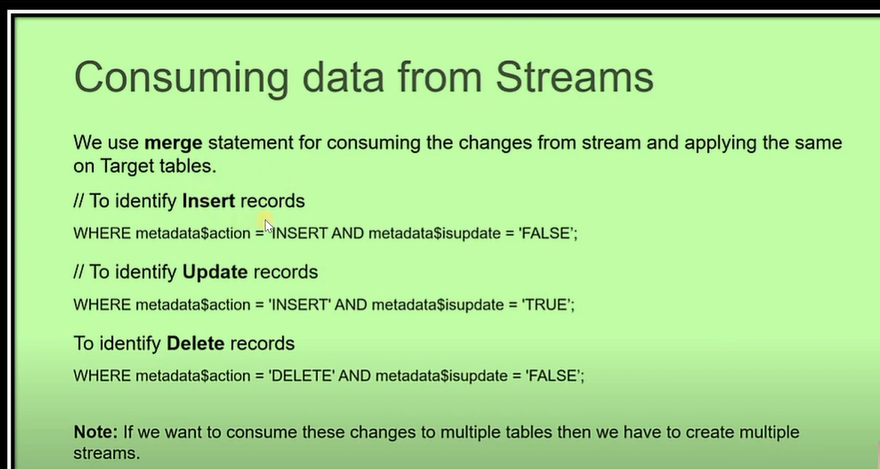
**1.METADATA$ACTION:** Indicates the DML operation (INSERT, DELETE) recorded.

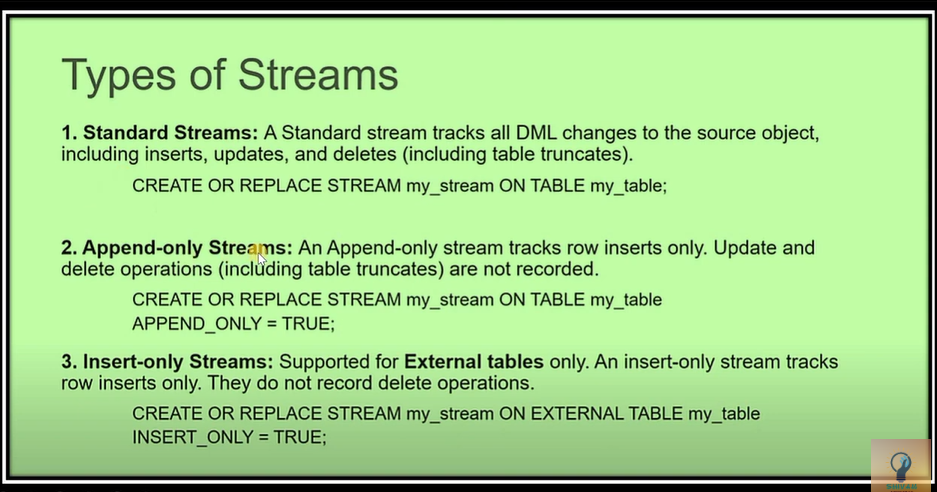
**2.METADATA$ISUPDATE:** Indicates whether the operation was part of an UPDATE statement. Updates to rows in the rows in the source object are represented as a pair of DELETE and

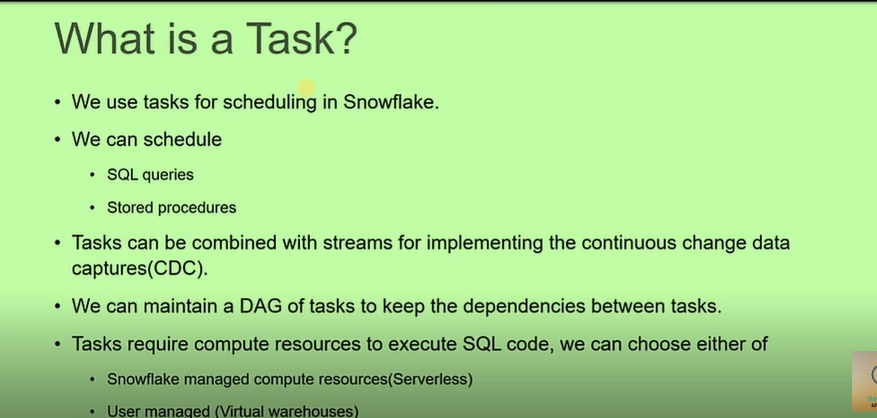
INSERT records in the stream with a metadata column **METADATA$ISUPDATE** values set to **TRUE**.

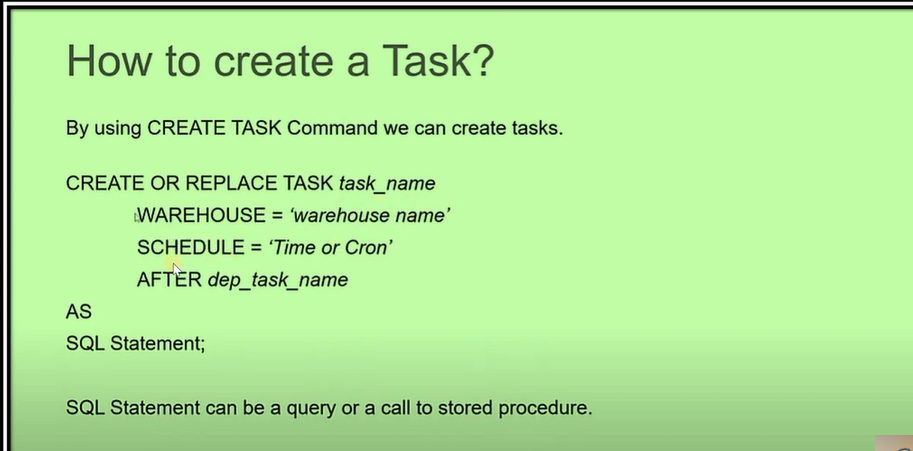
**3.METADATA$ROW\_ID:** Specifies the unique and immutable ID for the row, which can be used to track changes to specific rows over time.

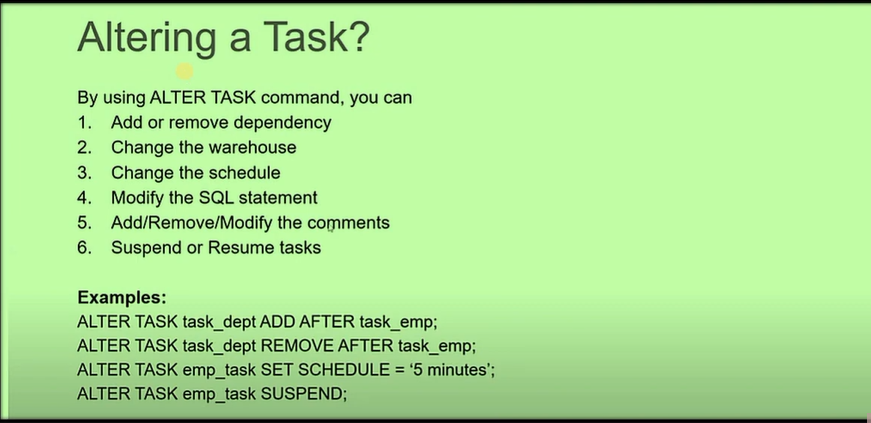
****

****

****

**What is Task?** ****

****

****

****

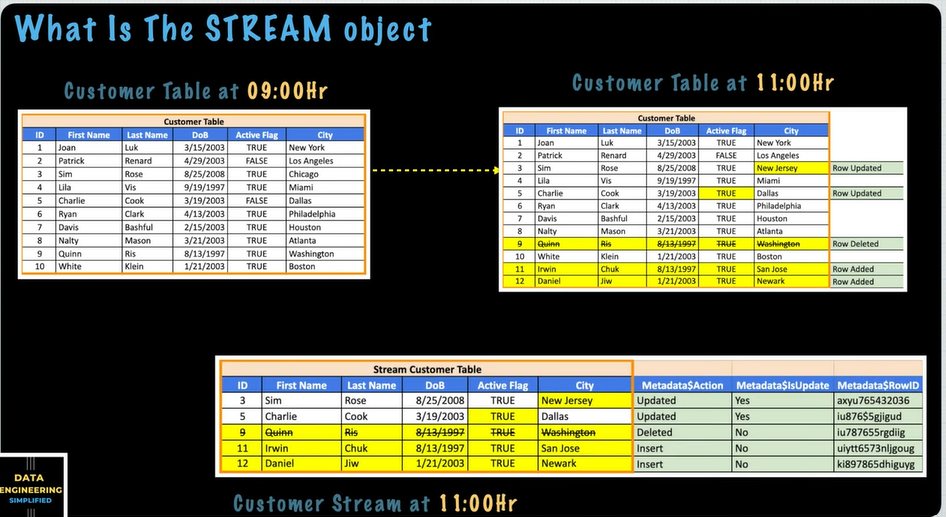
* **If you truncate and delete any records in the table, its capture.**

**CDC Tracker Metadata Fields**

1.**DML**(DATA MANIPULTION LANGUAGE ): insert, update, delete & TRUNCATE OPERATIONS (DML OPERATIONS)

2.Additional Metadata Columns Added to the source or base table

**Metadata$Action,Metadata$isupdate,Metadata$RowID**



**Type of Streams:**

**1.Append only streams**

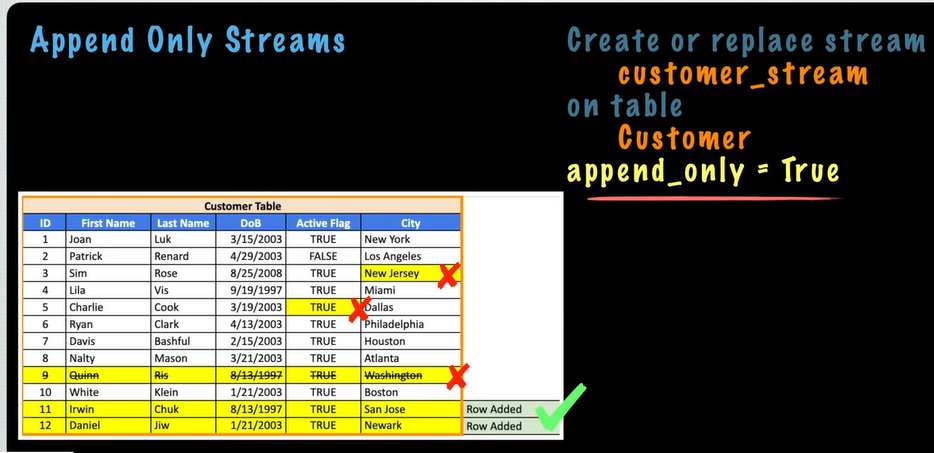
Create or replace stream

Customer \_stream

on table

customer

**append\_only =True**

****

**2.Insert only Streams**

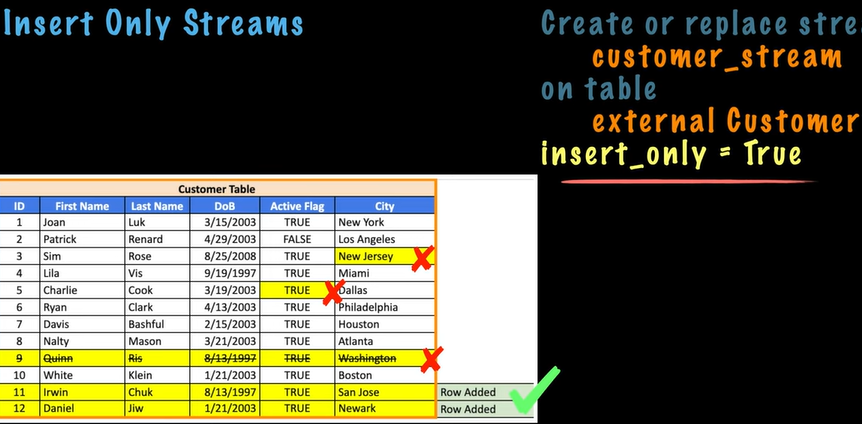
Create or replace stream

Customer \_stream

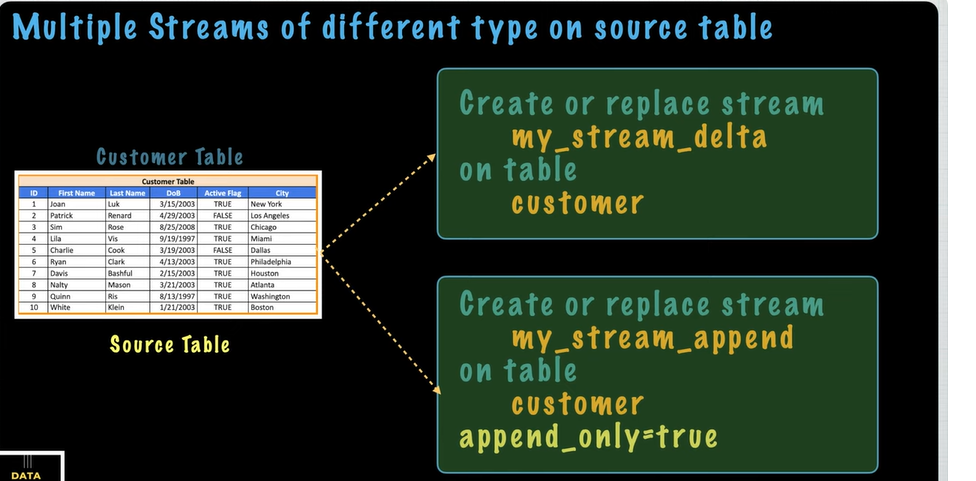
on table

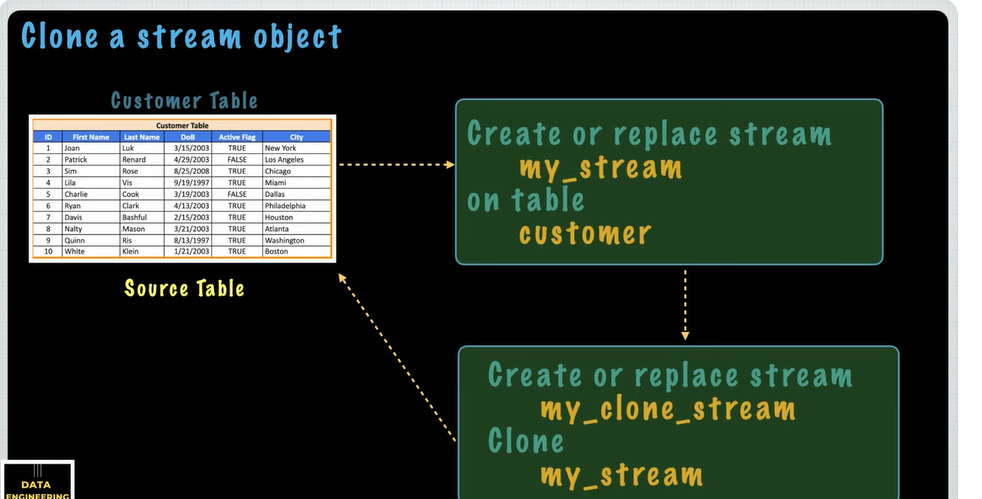
customer

**insert\_only =True**

****

**Multiple Streams on Source Table**

****

****

**Store Procedure**

A like functions, a store procedure is created once and can be executed many times.

A store procedure is created with a CREATE PROCEDURE command with ca call command.

* **You can write a stored procedure in one of the following languages:**
* Java (using snowspark)
* JavaScript
* Python (using Snowspark)
* Scala (using Snowspark)
* Writing Stored Procedures in Snowflake Scripting
* From a stored procedure, you can return a single value of (if you are using Snowflake Scripting) tabular data.

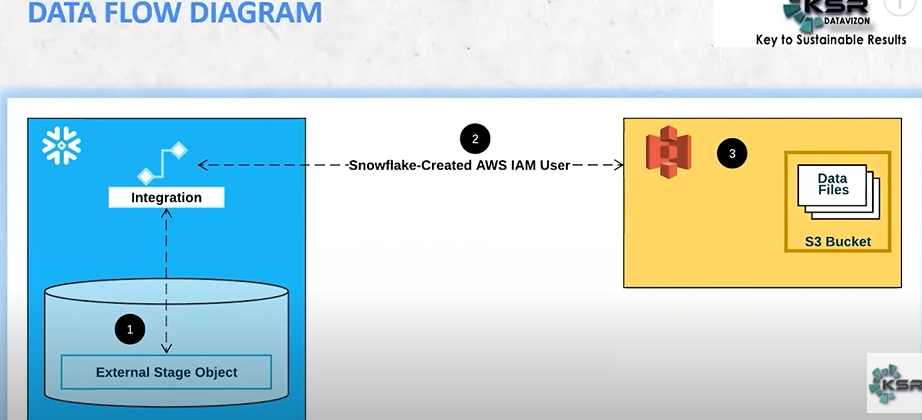
**Benefits of Store Procedures**

**Store procedures allow:**

* Procedural logic (branching: **if and else**) & looping: **while, for loop**) which straight SQL does not support.
* Error handling – **TRY-CATCH blocks (java this must be executed log error or either exception), RAISE\_ERROR function (customed error)**
* Dynamically creating a SQL statement and execute it.

(**Whatever code you want to give you can just give at the runtime** only means you don’t have to write your code in the program or script

**INTEGRATION AWS TO SNOWFLAKE:**

****

s3 create bucket(demo\_17012025) --🡪 upload file 🡪add files(customer\_info.csv) -🡪 click on customer file 🡪 copy s3 url (s3:// demo\_17012025/ customer\_info.csv

IAM (identity and access and management )

---policy ---.new policy ----default(S \_Permissions\_policy)—Json --- give me review policy -- demo\_170125) – create policy

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:GetObject",

"s3:GetObjectVersion",

"s3:DeleteObject",

"s3:DeleteObjectVersion"

],

"Resource": "arn:aws:s3:::<bucket>/<prefix>/\*"

},

{

"Effect": "Allow",

"Action": [

"s3:ListBucket",

"s3:GetBucketLocation"

],

"Resource": "arn:aws:s3:::<bucket>",

"Condition": {

"StringLike": {

"s3:prefix": [

"\*"

]

}

}

}

]

}

https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration

Role 🡪 AWS ACCOUNT -🡪 THIS ACCOUNT ID (UNIQUE ID) 🡪 EXTERNAL ID (ACCOUNT ID )-🡪NEXT -🡪

What policy you can attached you can select (demo\_170125) permsssions – create role – click on role

-🡪 Generate role ARM number url (copy and paste in notebook )

SF

1.store integration

2.file format

3.stage

**Integration**

**CREATE OR REPLACE STORAGE INTEGRATION my\_s3\_integration**

**TYPE = EXTERNAL\_STAGE**

**STORAGE\_PROVIDER = 'S3'**

**STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::123456789012:role/my-s3-role'**

**STORAGE\_ALLOWED\_LOCATIONS = ('s3://my-bucket/path/')**

**ENABLED = TRUE;**

CREATE STORAGE INTEGRATION <integration\_name>

TYPE = EXTERNAL\_STAGE

STORAGE\_PROVIDER = 'S3'

ENABLED = TRUE

STORAGE\_AWS\_ROLE\_ARN = '<iam\_role>'

STORAGE\_ALLOWED\_LOCATIONS = ('<protocol>://<bucket>/<path>/', '<protocol>://<bucket>/<path>/')

[ STORAGE\_BLOCKED\_LOCATIONS = ('<protocol>://<bucket>/<path>/', '<protocol>://<bucket>/<path>/') ]

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

CREATE STORAGE INTEGRATION s3\_int

TYPE = EXTERNAL\_STAGE

STORAGE\_PROVIDER = 'S3'

ENABLED = TRUE

STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::001234567890:role/myrole'

STORAGE\_ALLOWED\_LOCATIONS = ('\*')

STORAGE\_BLOCKED\_LOCATIONS = ('s3://mybucket1/mypath1/sensitivedata/', 's3://mybucket2/mypath2/sensitivedata/');

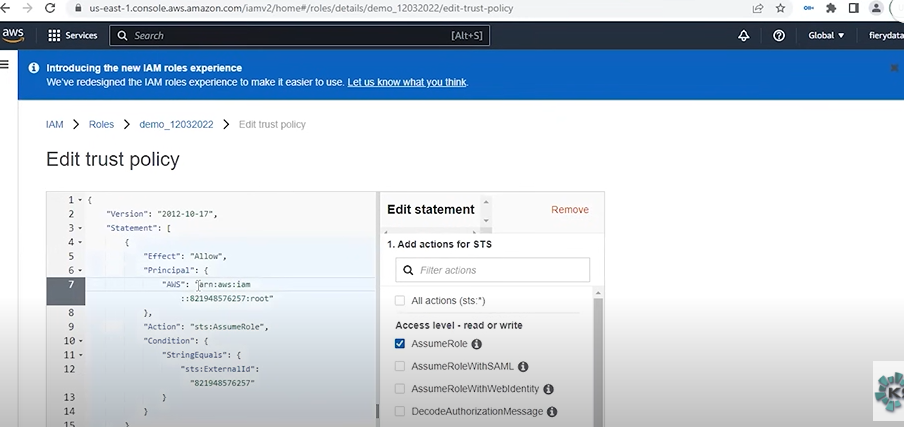
**Show integration.**

**Desc integration my\_s3\_integration**



Copy row 5 url (STORAGE\_AWS\_IAM\_USER\_ARN) ---GO TO ROLE – TRUST RELATIONSHIPS – EDIT TRUST POLICY ---

Copy row 7 url (STORAGE\_AWS\_EXTERNAL\_ID)



7. AWS : STORAGE\_AWS\_IAM\_USER\_ARN (URL)

12. sts: “EXTERNALID”:STORAGE\_AWS\_EXTERNAL\_ID -🡪 UPDATE POLICY

SNOWFLAKE

Grant usage on integration < store\_integration name > to role accountadmin;

**CREATE OR REPLACE FILE FORMAT demo\_format**

**TYPE = 'CSV'**

**FIELD\_DELIMITER = ','**

**SKIP\_HEADER = 1;**

https://docs.snowflake.com/en/user-guide/data-load-s3-create-stage

**CREATE OR REPLACE STAGE demo\_aws\_stage**

**URL = 's3://my-bucket/path/'**

**STORAGE\_INTEGRATION = my\_s3\_integration**

**FILE\_FORMAT =demo\_format or (TYPE = 'CSV' FIELD\_DELIMITER = ','**

**SKIP\_HEADER = 1;**

**OR**

**CREATE OR REPLACE STAGE demo\_aws\_stage**

**URL = 's3://demo\_17012025/orders/'**

**STORAGE\_INTEGRATION = my\_s3\_integration**

**FILE\_FORMAT =demo\_format;**

**list@<stage\_name>demo\_aws\_stage**

I want to remove customer\_info.csv file in s3 bucket

**remove@demo\_aws\_stage/customer\_info.csv**

**--- add files in s3 location : customer\_info.csv**

Create or replace temporary table demo\_customer\_info(

Id integer,

Last\_name string,

First\_name string ,

Company string,

Email string,

Workphone string,

Cellphone strig,

Streetaddress string,

City string,

Postalcode string

);

Select \* from demo\_customer\_info limit 10;

**We will try to load it from an external using external stage accessing the file which is available on AWS S3**

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/**

**File\_format =(format\_name=demo\_format);**

**--** Select \* from demo\_customer\_info limit 10;

**Scenario1:** there are 100 files available okay and you have toload all the 100 files so how can you load all the 100 files. If any file has failed you want to load remaining files.

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/**

**File\_format =(format\_name=demo\_format);**

On\_error = ‘skip\_file’; -- skip the whole file

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/**

**File\_format =(format\_name=demo\_format);**

On\_error = ‘continue’; 🡪 skips only the bad records and load rest of the records

means there are 100 records in a file and there are 2 bad records so what you can do is this option skips only the 2 bad records and rest of the 90 records will get loaded into your tables.

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/**

**File\_format =(format\_name=demo\_format);**

On\_error = ‘abort’; 🡪 right so any issue happened any bad record appeared all you need to do is you have to kill you have to stop your process.

**Scenario\_2:**

so what happens now if I try to load this data again okay let's try to understand if I try to load the data again the same data so I'm going to pick this file name okay I already loaded this data and we able to see the data over here right if I try to load the same data into a table for me this command shows and **not an error** it shows an information saying that it's processed with the **Zero** Records.

**list@demo\_aws\_stage/customer\_info.csv;**

okay the file is still existing but while when I was trying to load the data again from the same file it shows that I have able to I'm able to process only Zero Records from this file the reason is the copy into command is a one of the features in a snowflake it has a memory it makes sure that whatever the

files it has loaded right that all the file information will keep in this memory and it able to understand that next time if you try to load the same file it understands you're trying to

load and duplicate data and it will stop you saying that without no notifications okay it stops you saying that okay is you executed me, but I processed only Zero Records because this file has been already loaded okay so copy into command will maintain a

history for a certain time and within

**Again load the same data**

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/customer\_info.csv**

**File\_format =(format\_name=demo\_format);**

**Scenario\_3:**

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/customer\_info.csv**

**File\_format =(format\_name=demo\_format)**

**force=true; -- again it had loaded the data.**

**now check how many records got loaded.**

**list@demo\_aws\_stage/customer\_info.csv;**

**----** Select count(\*) demo\_customer\_info limit 10;

10 records

**Scenario\_4:**

**successful completion of my load okay I must remove the file** okay in that sense what you can do is there is an option called **purging** concept okay these are all the inbuilt functionalities inside a you know copy into command.

what this purge is equals to True does as soon as you successfully load the data into your table okay it understands saying that on successful completion, I must remove that file okay I have to remove that file from the stage so automatically. **if you remove the file from stage that gets removed from your you know AWS.**

**See the file is available from me**

[**list@demo\_aws\_stage/customer\_info.csv**](mailto:list@demo_aws_stage/customer_info.csv)**;**

**COPY INTO demo\_customer\_info**

**FROM @demo\_aws\_stage/**

**File\_format =(format\_name=demo\_format)**

**force=true purge=true;**

**I have 5 records**

**I have 10 records**

**----** Select count(\*) demo\_customer\_info limit 10;

**Run the purge command and loaded 5 records total 15 records 10+05 =15 records**

**If see**

[**list@demo\_aws\_stage/customer\_info.csv**](mailto:list@demo_aws_stage/customer_info.csv)**; --- there is no files.**

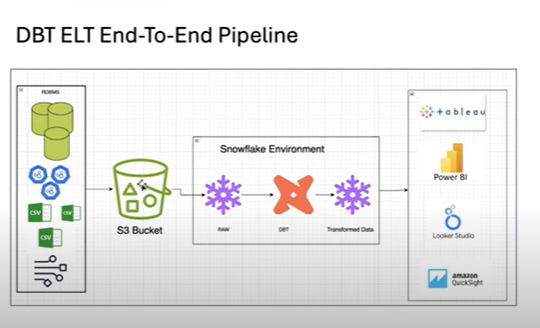
**And check it aws s3 bucket there is no files.**

**Some kind of functionalities**

**Validate - mode**

**Error if any columns mismatches**

**END-TO- END ELT PIPELINE PROJECT WITH DBT, SNOWFLAKE, TABLEAU**



* dbt is not an ingestion tool, you only use it to transform your data once you’ve loaded it your database or data warehouse.
* 3 python libraires

**1**.faker **2**. Boto3 **3.** Pandas

