**Standard Operating Procedure (SOP):**

**SCHEMA DETECTION AND EVOLUTION IN SNOWFLAKE**

OBJECTIVE:

To establish a structured process for detecting and evolving database schemas in Snowflake, ensuring data consistency, compatibility, and smooth operations as the data model evolves over time.

SCOPE: This SOP covers the following key areas:

**Schema Detection**: Identifying changes in schema or columns.

**Schema Evolution**: Managing schema changes such as adding, altering, or removing columns.

**Change Management**: Documenting and deploying schema changes in a controlled and automated manner.

Load Sample CSV and JSON Files into the Internal Stage Location

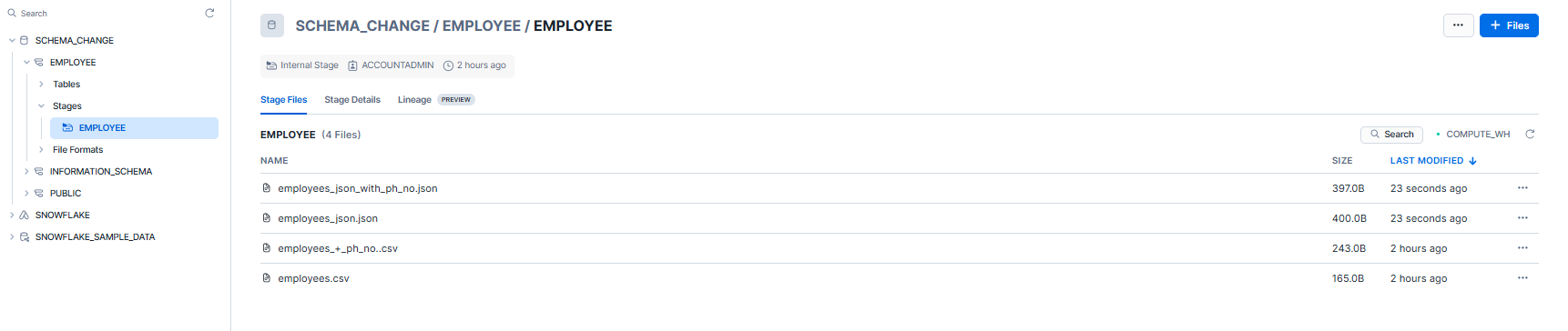
USE DATABASE SCHEMA\_CHANGE;

USE SCHEMA EMPLOYEE;

-- loaded csv and json in stg

SHOW STAGES;

LIST @"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE" ;



Here we loaded.

Case 1: CSV data

employees.csv: Table need to be updated with field phone number.

employees\_+\_ph\_no..csv: Updated table with phone number field.

Case 2: Json data

employees\_json: Table need to be updated with field phone number.

employees\_json\_with\_ph\_no: Updated table with phone number field.

**case 1**: CSV data.

Step 1: create file format.

CREATE OR REPLACE FILE FORMAT my\_csv\_format

TYPE = 'CSV'

FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'

PARSE\_HEADER = True

FIELD\_DELIMITER = ',’ ;

Step 2: Schema detection (Infer schema)

create employee\_table using infer\_schema.

CREATE OR REPLACE TABLE employee\_table

USING TEMPLATE (

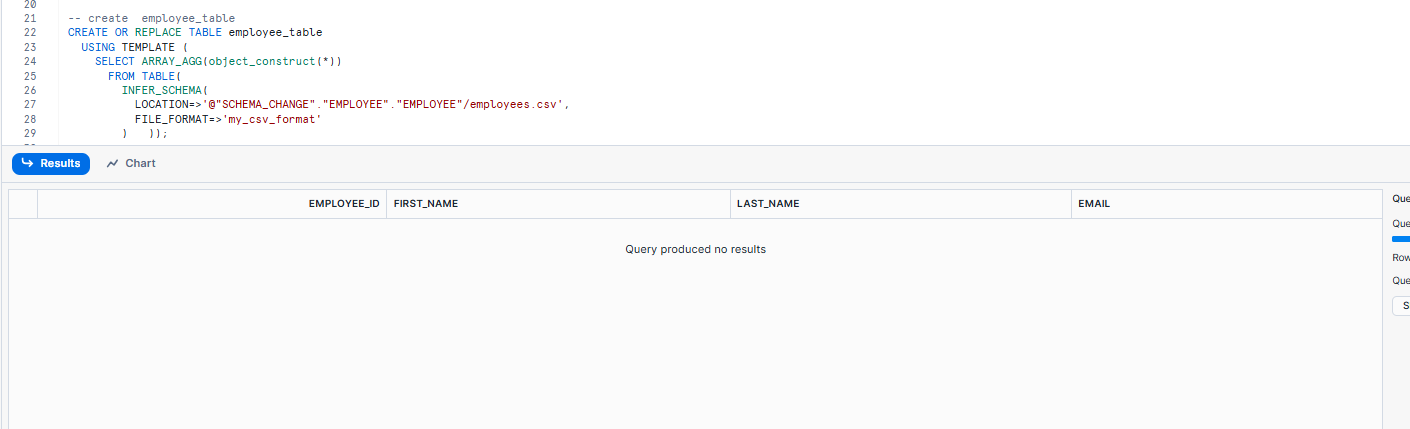
SELECT ARRAY\_AGG (object\_construct(\*))

FROM TABLE (

INFER\_SCHEMA (

LOCATION=>'@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/employees.csv’, FILE\_FORMAT=>'my\_csv\_format' ) ));

In this important step is to use INFER\_SCHEMA, if we use it no need to define structure it automatically creates.



Step 3: Schema Evolution (Enable Schema Evolution).

-- check enable schema evolution on the table is Y or N

show tables.

|  |  |
| --- | --- |
|  |  |

In the above two tables for Employee table which we are using in **case 1** showing **N** in Enable Schema Evolution make it **Y** by running the below code.

-- enable schema evolution on the table to Y

ALTER TABLE employee table SET ENABLE\_SCHEMA\_EVOLUTION = TRUE;

Show tables:

|  |  |
| --- | --- |
|  |  |

Step 4: Copy data from stage.

-- load data into employee\_table

COPY INTO employee\_table FROM

@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/employees.csv

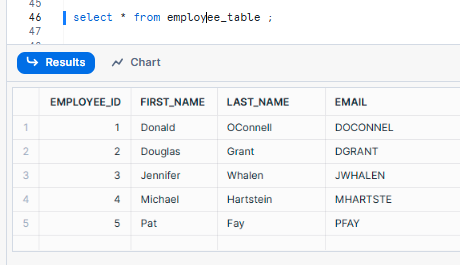
FILE\_FORMAT = (FORMAT\_NAME= 'my\_csv\_format' , error\_on\_column\_count\_mismatch=false)

MATCH\_BY\_COLUMN\_NAME=CASE\_INSENSITIVE ;

In this important step is to use

error\_on\_column\_count\_mismatch=false and MATCH\_BY\_COLUMN\_NAME=CASE\_INSENSITIVE

using the above will easily append changes in the fields name or if new field found.



Step 5: New field phone number append to the same table.

-- load data into employee\_table with phone number

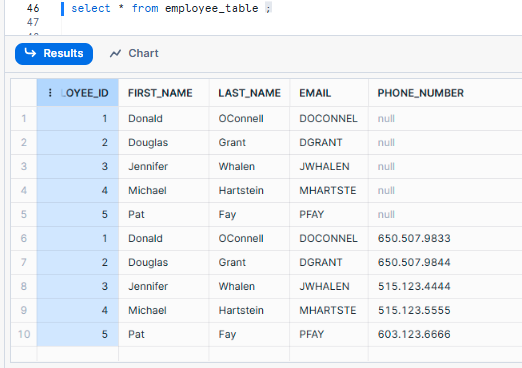
COPY INTO employee\_table FROM

@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/employees\_+\_ph\_no..csv

FILE\_FORMAT = (FORMAT\_NAME= 'my\_csv\_format' , error\_on\_column\_count\_mismatch=false )

MATCH\_BY\_COLUMN\_NAME=CASE\_INSENSITIVE ;

By running above code, the same table employee\_table will append new field as shown below.



Data here loaded as incremental.

**case 2**: JSON data.

Step 1: create file format.

-- Create a file format that sets the file type as Parquet.

CREATE OR REPLACE FILE FORMAT my\_json\_format

TYPE = json,

STRIP\_OUTER\_ARRAY=TRUE,

date\_format=auto,

time\_format=auto,

timestamp\_format=auto;

Step 2: Schema detection (Infer schema)

create employee\_table\_json using infer\_schema.

CREATE OR REPLACE TABLE employee\_table\_json

USING TEMPLATE (

SELECT ARRAY\_AGG(OBJECT\_CONSTRUCT(\*))

FROM TABLE(

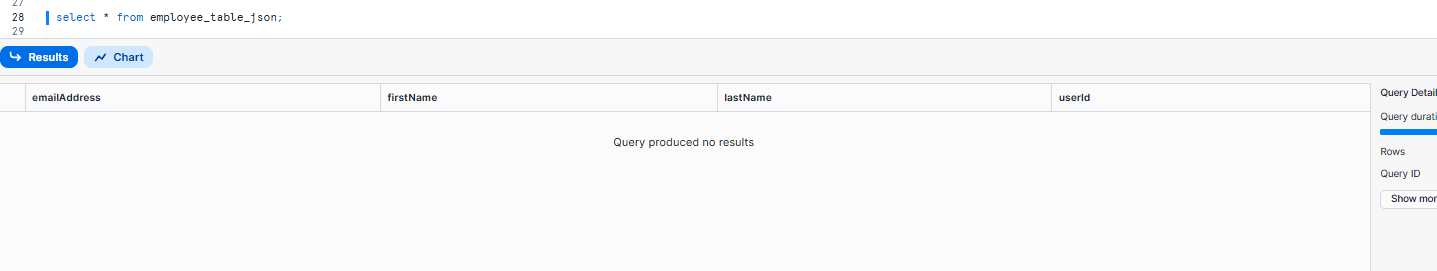
INFER\_SCHEMA(

LOCATION=>'@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/employees\_json.json',

FILE\_FORMAT=>'my\_json\_format'

) ));

In this important step is to use INFER\_SCHEMA, if we use it no need to define structure it automatically creates.



Step 3: Schema Evolution (Enable Schema Evolution).

-- check enable schema evolution on the table is Y or N

show tables.

|  |  |
| --- | --- |
|  |  |

In the above two tables for Employee table\_jason which we are using in **case 2** showing **Y** by running the below code.

-- enable schema evolution on the table to Y

ALTER TABLE employee\_table\_json SET ENABLE\_SCHEMA\_EVOLUTION = TRUE;

Step 4: Copy data from stage.

-- Add data to the employee\_table\_json

COPY INTO employee\_table\_json

FROM '@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/'

FILES=('employees\_json.json')

FILE\_FORMAT=my\_json\_format

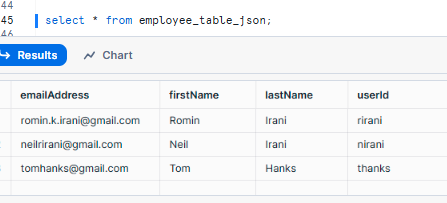
match\_by\_column\_name=case\_insensitive

purge=true;

In this important step is to use

MATCH\_BY\_COLUMN\_NAME=CASE\_INSENSITIVE and purge=true;

using the above will easily append changes in the fields name or if new field found.



Step 5: New field phone number append to the same table.

-- update field to the employee\_table\_json

COPY INTO employee\_table\_json

FROM '@"SCHEMA\_CHANGE"."EMPLOYEE"."EMPLOYEE"/'

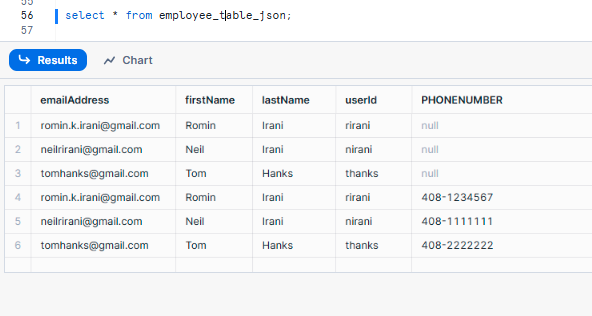
FILES=('employees\_json\_with\_ph\_no.json')

FILE\_FORMAT=my\_json\_format

match\_by\_column\_name=case\_insensitive

purge=true;

By running above code, the same table employee\_table\_json will append new field as shown below.



Data here loaded as incremental.