**28-01-2025**

SELECT

postal\_code FROM

onpoint\_id.history\_day ORDER BY

postal\_code,country,date\_valid\_std

;

-- Set role context

USE ROLE sysadmin;

-- Set warehouse context

USE WAREHOUSE compute\_wh;

-- Create a database

CREATE [OR REPLACE] DATABASE [IF NOT EXISTS] weather;

USE DATABASE SNOWFLAKE;

select \* from SNOWFLAKE\_SAMPLE\_DATA.INFORMATION\_SCHEMA.APPLICABLE\_ROLES;

**29-01-2025**

SELECT DATEADD(day,1,CURRENT\_DATE()) tomorrow, CURRENT\_DATE() today;

SELECT DATEADD(month,1,'2025-01-15' :: date) tomorrow, CURRENT\_DATE() today;

SELECT DATE\_TRUNC(day,DATEADD(year,-2,CURRENT\_DATE())) date2;

SELECT LAST\_DAY(DATEADD(week,2 + CAST())

**31-01-2025**

SELECT

DATE\_TRUNC(year,DATEADD(year,-1,CURRENT\_DATE())) AS ref\_timestamp,

WEEKISO(ref\_timestamp) as ref1,

WEEKISO(ref\_timestamp) != 1 as ref2,

CAST(WEEKISO(ref\_timestamp) != 1 AS INTEGER) AS ref3, --converts the data type of an expression to the specified data type

2 + CAST(WEEKISO(ref\_timestamp) != 1 AS INTEGER) AS ref4,

DATEADD(week,2 + CAST(WEEKISO(ref\_timestamp) != 1 AS INTEGER), ref\_timestamp) AS ref5,

LAST\_DAY(DATEADD(week,2 + CAST(WEEKISO(ref\_timestamp) != 1 AS INTEGER),ref\_timestamp),week) AS end\_week,

DATEADD(day, day\_num - 7, end\_week) AS date\_valid\_std

FROM

(

SELECT

ROW\_NUMBER() OVER (ORDER BY SEQ1()) AS day\_num

FROM

TABLE(GENERATOR(rowcount => 7))

)

--compares two dates and returns the difference

-- Snowflake Stages are locations where data files are stored (“staged”) which helps in loading data into and unloading data out of database tables. The stage locations could be internal or external to Snowflake environment.

-- A Snowflake file format is a named database object that encapsulates information about a data file

**04-02-2025**

CREATE OR REPLACE FILE FORMAT my\_csv\_format

TYPE = CSV

FIELD\_DELIMITER = ','

SKIP\_HEADER = 1

-- NULL\_IF = ('NULL', 'null')

-- EMPTY\_FIELD\_AS\_NULL = true

-- COMPRESSION = gzip;

CREATE OR REPLACE STAGE MY\_STAGE;

LIST @MY\_STAGE;

-- IN STAGE WE CAN ACCESS DATA FROM $1 = COLUMN1

SELECT $1 FROM @MY\_STAGE;

SELECT $1,$2,$3,$4,$5,$6,$7 FROM @MY\_STAGE(FILE\_FORMAT=>MY\_CSV\_FORMAT)

CREATE TABLE trip\_data (

tripduration INT,

starttime TIMESTAMP,

stoptime TIMESTAMP,

start\_station\_id INT,

start\_station\_name STRING,

start\_station\_latitude FLOAT,

start\_station\_longitude FLOAT,

end\_station\_id INT,

end\_station\_name STRING,

end\_station\_latitude FLOAT,

end\_station\_longitude FLOAT,

bikeid INT,

usertype STRING,

birth\_year INT,

gender INT

);

COPY INTO trip\_data

FROM @MY\_STAGE

FILE\_FORMAT = (FORMAT\_NAME = MY\_CSV\_FORMAT);

SELECT \* FROM trip\_data;

**10-02-2025**

create file format new\_csv

type = { CSV | JSON | AVRO | ORC | PARQUET | XML }

-- [ formatTypeOptions ]

-- comment = '<comment>'

create or replace file format new\_csv

type=csv

field\_delimiter=','

skip\_header=1

FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'

list @NEW\_STAGE;

select $1,$2,$3,$4 from @new\_stage;

select $1,$2,$3,$4 from @new\_stage(file\_format=>new\_csv)

with d as (

SELECT ROW\_NUMBER() over(partition by 1 order by 1) as REF\_ID, S.$1, S.$2,S.$3,S.$4,S.$5,S.$6,S.$7,S.$8,S.$9,S.$10,

S.$11, S.$12,S.$13,S.$14,S.$15,S.$16,S.$17,S.$18,S.$19,S.$20, S.$21, S.$22,S.$23,S.$24,S.$25,S.$26,S.$27,S.$28,S.$29,S.$30,

S.$31, S.$32,S.$33,S.$34,

case when trim(S.$1)='Chart ID :' then S.$3 end as Chart\_ID,

case when S.$1='Client Name :' then S.$3 end as Client\_Name,

case when S.$1='Start Date :' then S.$3 end as Start\_Date,

case when S.$19='End Date :' then S.$24 end as End\_Date,

case when S.$32 is null then 'WPT- Williamsport' end as Location

from @new\_stage(file\_format =>new\_csv) As S),

Y as (

select

REF\_ID,

nvl(Chart\_ID, Lag(Chart\_ID) ignore nulls over (order by Ref\_id)) as Chart\_ID,

nvl(Client\_Name, Lag(Client\_Name) ignore nulls over (order by Ref\_id)) as Client\_Name,

nvl(Start\_Date, lag (Start\_Date) ignore nulls over (order by Ref\_id)) as Start\_Date,

nvl(End\_Date, Lag (End\_Date) ignore nulls over (order by Ref\_id)) as End\_Date,

Location,

case when $2 is not null and $2 not in ('Task','Times') then $1 end as Task\_id,

case when $2 is not null and $2 not in ('Task', 'Times') then $2 end as Task,

case when $2 is not null and $2 not in ('Task', 'Times') then $6 end as Time,

case when $2 is not null and $2 not in ('Task', 'Times') then $9 end as Sun,

case when $2 is not null and $2 not in ('Task', 'Times') then $12 end as Mon,

case when $2 is not null and $2 not in ('Task', 'Times') then $16 end as Tue,

case when $2 is not null and $2 not in ('Task', 'Times') then $19 end as Wed,

case when $2 is not null and $2 not in ('Task', 'Times') then $23 end as Thur,

case when $2 is not null and $2 not in ('Task', 'Times') then $27 end as Fri,

case when $2 is not null and $2 not in ('Task', 'Times') then $30 end as Sat,

from d)

SELECT \* FROM Y WHERE Time IS NOT NULL;

create or replace temp table my\_table as

with d as (

SELECT ROW\_NUMBER() over(partition by 1 order by 1) as REF\_ID, S.$1, S.$2,S.$3,S.$4,S.$5,S.$6,S.$7,S.$8,S.$9,S.$10,

S.$11, S.$12,S.$13,S.$14,S.$15,S.$16,S.$17,S.$18,S.$19,S.$20, S.$21, S.$22,S.$23,S.$24,S.$25,S.$26,S.$27,S.$28,S.$29,S.$30,

S.$31, S.$32,S.$33,S.$34,

case when trim(S.$1)='Chart ID :' then S.$3 end as Chart\_ID,

case when S.$1='Client Name :' then S.$3 end as Client\_Name,

case when S.$1='Start Date :' then S.$3 end as Start\_Date,

case when S.$19='End Date :' then S.$24 end as End\_Date,

case when S.$32 is null then 'WPT- Williamsport' end as Location

from @new\_stage(file\_format =>new\_csv) As S),

Y as (

select

REF\_ID,

nvl(Chart\_ID, Lag(Chart\_ID) ignore nulls over (order by Ref\_id)) as Chart\_ID,

nvl(Client\_Name, Lag(Client\_Name) ignore nulls over (order by Ref\_id)) as Client\_Name,

nvl(Start\_Date, lag (Start\_Date) ignore nulls over (order by Ref\_id)) as Start\_Date,

nvl(End\_Date, Lag (End\_Date) ignore nulls over (order by Ref\_id)) as End\_Date,

Location,

case when $2 is not null and $2 not in ('Task','Times') then $1 end as Task\_id,

case when $2 is not null and $2 not in ('Task', 'Times') then $2 end as Task,

case when $2 is not null and $2 not in ('Task', 'Times') then $6 end as Time,

case when $2 is not null and $2 not in ('Task', 'Times') then $9 end as Sun,

case when $2 is not null and $2 not in ('Task', 'Times') then $12 end as Mon,

case when $2 is not null and $2 not in ('Task', 'Times') then $16 end as Tue,

case when $2 is not null and $2 not in ('Task', 'Times') then $19 end as Wed,

case when $2 is not null and $2 not in ('Task', 'Times') then $23 end as Thur,

case when $2 is not null and $2 not in ('Task', 'Times') then $27 end as Fri,

case when $2 is not null and $2 not in ('Task', 'Times') then $30 end as Sat,

from d)

SELECT \* FROM Y WHERE Time IS NOT NULL;

create view view1 as

with d as (

SELECT ROW\_NUMBER() over(partition by 1 order by 1) as REF\_ID, S.$1, S.$2,S.$3,S.$4,S.$5,S.$6,S.$7,S.$8,S.$9,S.$10,

S.$11, S.$12,S.$13,S.$14,S.$15,S.$16,S.$17,S.$18,S.$19,S.$20, S.$21, S.$22,S.$23,S.$24,S.$25,S.$26,S.$27,S.$28,S.$29,S.$30,

S.$31, S.$32,S.$33,S.$34,

case when trim(S.$1)='Chart ID :' then S.$3 end as Chart\_ID,

case when S.$1='Client Name :' then S.$3 end as Client\_Name,

case when S.$1='Start Date :' then S.$3 end as Start\_Date,

case when S.$19='End Date :' then S.$24 end as End\_Date,

case when S.$32 is null then 'WPT- Williamsport' end as Location

from @new\_stage(file\_format =>new\_csv) As S),

Y as (

select

REF\_ID,

nvl(Chart\_ID, Lag(Chart\_ID) ignore nulls over (order by Ref\_id)) as Chart\_ID,

nvl(Client\_Name, Lag(Client\_Name) ignore nulls over (order by Ref\_id)) as Client\_Name,

nvl(Start\_Date, lag (Start\_Date) ignore nulls over (order by Ref\_id)) as Start\_Date,

nvl(End\_Date, Lag (End\_Date) ignore nulls over (order by Ref\_id)) as End\_Date,

Location,

case when $2 is not null and $2 not in ('Task','Times') then $1 end as Task\_id,

case when $2 is not null and $2 not in ('Task', 'Times') then $2 end as Task,

case when $2 is not null and $2 not in ('Task', 'Times') then $6 end as Time,

case when $2 is not null and $2 not in ('Task', 'Times') then $9 end as Sun,

case when $2 is not null and $2 not in ('Task', 'Times') then $12 end as Mon,

case when $2 is not null and $2 not in ('Task', 'Times') then $16 end as Tue,

case when $2 is not null and $2 not in ('Task', 'Times') then $19 end as Wed,

case when $2 is not null and $2 not in ('Task', 'Times') then $23 end as Thur,

case when $2 is not null and $2 not in ('Task', 'Times') then $27 end as Fri,

case when $2 is not null and $2 not in ('Task', 'Times') then $30 end as Sat,

from d)

SELECT \* FROM Y WHERE Time IS NOT NULL;

select \* from view1;

create or replace table practice\_test\_data1 as

with d as (

SELECT ROW\_NUMBER() over(partition by 1 order by 1) as REF\_ID, S.$1, S.$2,S.$3,S.$4,S.$5,S.$6,S.$7,S.$8,S.$9,S.$10,

S.$11, S.$12,S.$13,S.$14,S.$15,S.$16,S.$17,S.$18,S.$19,S.$20, S.$21, S.$22,S.$23,S.$24,S.$25,S.$26,S.$27,S.$28,S.$29,S.$30,

S.$31, S.$32,S.$33,S.$34,

case when trim(S.$1)='Chart ID :' then S.$3 end as Chart\_ID,

case when S.$1='Client Name :' then S.$3 end as Client\_Name,

case when S.$1='Start Date :' then S.$3 end as Start\_Date,

case when S.$19='End Date :' then S.$24 end as End\_Date,

case when S.$32 is null then 'WPT- Williamsport' end as Location

from @new\_stage(file\_format =>new\_csv) As S),

Y as (

select

REF\_ID,

nvl(Chart\_ID, Lag(Chart\_ID) ignore nulls over (order by Ref\_id)) as Chart\_ID,

nvl(Client\_Name, Lag(Client\_Name) ignore nulls over (order by Ref\_id)) as Client\_Name,

nvl(Start\_Date, lag (Start\_Date) ignore nulls over (order by Ref\_id)) as Start\_Date,

nvl(End\_Date, Lag (End\_Date) ignore nulls over (order by Ref\_id)) as End\_Date,

Location,

case when $2 is not null and $2 not in ('Task','Times') then $1 end as Task\_id,

case when $2 is not null and $2 not in ('Task', 'Times') then $2 end as Task,

case when $2 is not null and $2 not in ('Task', 'Times') then $6 end as Time,

case when $2 is not null and $2 not in ('Task', 'Times') then $9 end as Sun,

case when $2 is not null and $2 not in ('Task', 'Times') then $12 end as Mon,

case when $2 is not null and $2 not in ('Task', 'Times') then $16 end as Tue,

case when $2 is not null and $2 not in ('Task', 'Times') then $19 end as Wed,

case when $2 is not null and $2 not in ('Task', 'Times') then $23 end as Thur,

case when $2 is not null and $2 not in ('Task', 'Times') then $27 end as Fri,

case when $2 is not null and $2 not in ('Task', 'Times') then $30 end as Sat,

from d)

SELECT \* FROM Y WHERE Time IS NOT NULL;

select \* from practice\_test\_data1;

alter table practice\_test\_data1 add column inserted\_data timestamp;

update practice\_test\_data1

set inserted\_data = current\_timestamp;

create table first\_table (

REF\_ID INT,

CHART\_ID VARCHAR,

CLIENT\_NAME STRING,

START\_DATE TIMESTAMP,

END\_DATE TIMESTAMP,

LOCATION STRING,

TASK\_ID INT,

TASK STRING,

TIME INT,

SUN STRING,

MON STRING,

TUE STRING,

WED STRING,

THUR STRING,

FRI STRING,

SAT STRING,

INSERTED\_DATA TIMESTAMP

);

create or replace procedure my\_first\_procedure()

RETURNS varchar not null

LANGUAGE sql

as

$$

begin

insert overwrite into first\_table

select \* from practice\_test\_data1;

return 'Success';

end;

$$;

call my\_first\_procedure()

select \* from first\_table;

create or replace procedure my\_first\_procedure2()

RETURNS varchar not null

LANGUAGE sql

as

$$

begin

insert into first\_table

select \* from practice\_test\_data1;

return 'Success';

end;

$$;

call my\_first\_procedure2()

select \* from first\_table;

select\*from first\_table BEFORE(STATEMENT=>'01ba7769-3201-71ff-000b-c30a000250ba');

select \* from first\_table;

select\*from first\_table AT(OFFSET=> -60\*1);

select\*from first\_table at(Statement=>'01ba7775-3201-71ff-000b-c30a000251ae');

select\*from first\_table AT(TIMESTAMP=>'2025-02-17 19:40:43'::TIMESTAMP\_LTZ);

SELECT CURRENT\_TIMESTAMP;

**19-02-2025**

create stage internal;

list @internal;

create stage s3\_stage

URL='s3://mybibbbucket56/cars.csv'

FILE\_FORMAT=(TYPE='CSV')

CREDENTIALS=(AWS\_KEY\_ID='ASIAUW2KEDUP4DACJWI6'

AWS\_SECRET\_KEY='qdkSs18Oen2vqzsMPLZe0YmxBkoxe6GpvpAW0EIS'

AWS\_TOKEN='');

list @s3\_stage;

select $1,$2 from @s3\_stage;

**04/03/2025**

* **First create the database, schema, stage& file format**
* **Then load file in the stage**

CREATE OR REPLACE FILE FORMAT JSON\_FILE\_FORMAT

TYPE = JSON

STRIP\_OUTER\_ARRAY = TRUE -- If JSON contains an outer array, this flattens it

-- FIELD\_DELIMITER = ','

-- SKIP\_HEADER = 1

-- NULL\_IF = ('NULL', 'null')

-- EMPTY\_FIELD\_AS\_NULL = true

-- COMPRESSION = gzip;

LIST @JSON\_STAGE

SELECT $1 FROM @JSON\_STAGE;

CREATE TABLE JSON\_TABLE(DATA VARIANT);

COPY INTO JSON\_TABLE FROM @JSON\_STAGE;

COPY INTO JSON\_TABLE FROM @JSON\_STAGE FILE\_FORMAT=(TYPE='JSON' STRIP\_OUTER\_ARRAY=TRUE);

SELECT \* FROM JSON\_TABLE;

SELECT

data:id::STRING AS donut\_id,

data:name::STRING AS donut\_name,

data:ppu::FLOAT AS price\_per\_unit,

batter.value:id::STRING AS batter\_id,

batter.value:type::STRING AS batter\_type,

topping.value:id::STRING AS topping\_id,

topping.value:type::STRING AS topping\_type

FROM

JSON\_TABLE,

LATERAL FLATTEN(input => parse\_json(data)) AS main\_node, -- Add comma before new LATERAL FLATTEN

LATERAL FLATTEN(input => main\_node.value:batter) AS batter,

LATERAL FLATTEN(input => main\_node.this:topping) AS topping;

**-- 24-03-2025**

-- Type 1 SCD (Overwriting)

-- The first possible type of SCD action is overwriting. Here, dimension values are overwritten by new values

-- Type 2 SCD (Row versioning)

-- The second type of SCD action is row versioning. In this type 2 SCD, when values for a current record change, the current record is marked as closed, and a new record gets inserted. Then, there will be 2 records associated with Mike in the updated table, but only the latest version will be marked “open”.

-- 3. Type 3 SCD (Adding previous value column)

-- The third common SCD type is adding a previous value column. Here, the last and current versions are maintained in a single row. In the customer MIke moving address example, we would copy Mike’s ‘Current City’ record into the ‘Previous City’ and overwrite the Current City record with the new city.

-- Other Types

-- In addition to these three types of SCDs, there are also Type 4 and Type 6. In Type 4, the dimension table has the latest value while its history is maintained in a separate table. In Type 6, a combination of Type 1, 2 & 3 are used to track changes in dimension. Generally, Type 6 is adopted in scenarios where multiple parts of a record are slowly changing dimensions, but using multiple implementations of a single type could lead to issues with rapid inflation of table size.

**Assignment**

-- 1. Create a file format to handle CSV files, specifying things like the delimiter, whether to skip headers, and other parameters.

CREATE OR REPLACE FILE FORMAT assignment\_csv

TYPE = 'CSV'

FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"'

SKIP\_HEADER = 1

FIELD\_DELIMITER = ','

NULL\_IF = ('NULL', '')

EMPTY\_FIELD\_AS\_NULL = TRUE;

list @ASSIGNMENT\_STAGE1;

CREATE STAGE ASSIGNMENT\_STAGE1

FILE\_FORMAT = assignment\_csv;

CREATE OR REPLACE TABLE my\_csv\_table (

col1 STRING,

col2 STRING,

col3 STRING,

col4 STRING

);

COPY INTO my\_csv\_table

FROM @assignment\_stage1

FILE\_FORMAT = (FORMAT\_NAME = assignment\_csv);

SELECT \* FROM my\_csv\_table;

CREATE OR REPLACE FILE FORMAT assignment\_csv

TYPE = 'CSV';

show stages;

select $1 from @json\_stage;

-- 3. Create a File Format for JSON and Load Data into a Snowflake Table

CREATE OR REPLACE TABLE my\_json\_table (

data VARIANT

);

x

select $1, $2, $3,$4,$5,$6 from @assignment\_stage;

-- Load Data into the Table

COPY INTO my\_json\_table

FROM @JSON\_STAGE

FILE\_FORMAT = (FORMAT\_NAME = my\_json\_format);

DESC TABLE my\_json\_table;

SELECT $1 FROM @JSON\_STAGE (FILE\_FORMAT => my\_json\_format);

select \* from my\_json\_table;

-- 4. Check Available Files in a Stage

list @ASSIGNMENT\_STAGE;

list @JSON\_STAGE;

-- Select all data from a table

SELECT \* FROM MY\_JSON\_TABLE;

-- Select specific columns

-- SELECT 1, 2 FROM MY\_JSON\_TABLE;

SELECT COUNT(\*) FROM MY\_JSON\_TABLE;

SELECT data FROM MY\_JSON\_TABLE LIMIT 1;

SELECT data FROM MY\_JSON\_TABLE;

SELECT

value:category::STRING AS category,

value:price::FLOAT AS price,

value:product\_id::INT AS product\_id,

value:product\_name::STRING AS product\_name,

value:stock\_quantity::INT AS stock\_quantity

FROM MY\_JSON\_TABLE,

LATERAL FLATTEN(input => data);

-- WHERE value:category = 'Electronics';

SELECT \* FROM MY\_JSON\_TABLE;

-- WHERE data:column1 = 'value';

CREATE OR REPLACE TABLE my\_fresh\_json (

category STRING,

price FLOAT,

product\_id INT,

stock\_quantity INT

);

CREATE OR REPLACE PROCEDURE load\_csv\_to\_table()

RETURNS VARCHAR

LANGUAGE SQL

AS

$$

BEGIN

-- Create table if not exists

CREATE TABLE IF NOT EXISTS my\_csv\_table (

COL1 STRING,

COL2 STRING,

COL3 STRING,

COL4 STRING

);

-- Load data from internal stage

COPY INTO my\_csv\_table

FROM @assignment\_stage1

FILE\_FORMAT = (FORMAT\_NAME = assignment\_csv);

RETURN 'CSV Data Loaded Successfully!';

END;

$$;

CALL load\_csv\_to\_table();

-- 6. Prepare the simple insert procedures for the given data in snowflake

CREATE OR REPLACE PROCEDURE MY\_PROCEDURE()

RETURNS VARCHAR

LANGUAGE SQL

AS

$$

BEGIN

-- Insert JSON data correctly

INSERT INTO my\_fresh\_json (category, price, product\_id, stock\_quantity)

SELECT

value:category::STRING AS category,

value:price::FLOAT AS price,

value:product\_id::INT AS product\_id,

value:stock\_quantity::INT AS stock\_quantity

FROM MY\_JSON\_TABLE,

LATERAL FLATTEN(input => data);

RETURN 'Data Inserted Successfully';

END;

$$;

call MY\_PROCEDURE();

**05-03-2025**

CREATE OR REPLACE FILE FORMAT JSON\_NEW\_FILE\_FORMAT

TYPE = JSON

STRIP\_OUTER\_ARRAY = TRUE

LIST @JSON\_STAGE

CREATE TABLE JSON\_NEW\_TABLE(DATA VARIANT);

COPY INTO JSON\_NEW\_TABLE FROM @JSON\_STAGE;

COPY INTO JSON\_NEW\_TABLE FROM @JSON\_STAGE FILE\_FORMAT=(TYPE='JSON' STRIP\_OUTER\_ARRAY=TRUE);

SELECT \* FROM JSON\_NEW\_TABLE;

CREATE PROCEDURE MY\_PROCEDURE(

RETURN VARCHAR

LANGUAGE SQL

AS

$$

INSERT INTO JSON\_TAB

SELECT

$$

)

SELECT

data:id::STRING AS donut\_id,

data:name::STRING AS donut\_name,

data:ppu::FLOAT AS price\_per\_unit,

batter.value:id::STRING AS batter\_id,

batter.value:type::STRING AS batter\_type,

topping.value:id::STRING AS topping\_id,

topping.value:type::STRING AS topping\_type

FROM

JSON\_NEW\_TABLE,

LATERAL FLATTEN(input => parse\_json(data)) AS main\_node, -- Add comma before new LATERAL FLATTEN

LATERAL FLATTEN(input => main\_node.value:batter) AS batter,

LATERAL FLATTEN(input => main\_node.this:topping) AS topping;

create table j\_table(

id int,

type varchar,

name varchar,

ppu float,

batters\_id string,

batters\_type string,

topping\_id string,

topping\_type string);

CREATE OR REPLACE PROCEDURE my\_json\_procedure()

RETURNS STRING NOT NULL

LANGUAGE SQL

AS

$$

-- BEGIN

INSERT INTO j\_table

SELECT

data:id::INT AS id,

data:type::STRING AS type,

data:name::STRING AS name,

data:ppu::FLOAT AS ppu,

batter.value:id::STRING AS batters\_id,

batter.value:type::STRING AS batters\_type,

topping.value:id::STRING AS topping\_id,

topping.value:type::STRING AS topping\_type

FROM JSON\_NEW\_TABLE,

LATERAL FLATTEN(input => data:batters.batter) AS batter,

LATERAL FLATTEN(input => data:topping) AS topping;

-- RETURN 'Success';

-- END;

$$;

call my\_json\_procedure()

select \* from j\_table;

CREATE TASK MY\_INSERT\_TASK

WAREHOUSE = COMPUTE\_WH

SCHEDULE = "1 MINUTES"

AS

CALL my\_json\_procedure();

SHOW TASKS;

ALTER TASK MY\_INSERT\_TASK RESUME;

ALTER TASK MY\_INSERT\_TASK SUSPEND;

SELECT \* FROM J\_TABLE;

TRUNCATE J\_TABLE

-- 21/03/2025

CREATE OR REPLACE STREAM MY\_STREAM ON TABLE J\_TABLE;

SELECT \* FROM MY\_STREAM;

SELECT \* EXCLUDE METADATA$ROW\_ID FROM MY\_STREAM;

**04-03-2025**

CREATE OR REPLACE FILE FORMAT JSON\_FILE\_FORMAT

TYPE = JSON

STRIP\_OUTER\_ARRAY = TRUE -- If JSON contains an outer array, this flattens it

-- FIELD\_DELIMITER = ','

-- SKIP\_HEADER = 1

-- NULL\_IF = ('NULL', 'null')

-- EMPTY\_FIELD\_AS\_NULL = true

-- COMPRESSION = gzip;

LIST @JSON\_STAGE

SELECT $1 FROM @JSON\_STAGE;

CREATE TABLE JSON\_TABLE(DATA VARIANT); --also use varchar

COPY INTO JSON\_TABLE FROM @JSON\_STAGE;

COPY INTO JSON\_TABLE FROM @JSON\_STAGE FILE\_FORMAT=(TYPE='JSON' STRIP\_OUTER\_ARRAY=TRUE);

SELECT \* FROM JSON\_TABLE;

SELECT

data:id::STRING AS donut\_id,

data:name::STRING AS donut\_name,

data:ppu::FLOAT AS price\_per\_unit,

batter.value:id::STRING AS batter\_id,

batter.value:type::STRING AS batter\_type,

topping.value:id::STRING AS topping\_id,

topping.value:type::STRING AS topping\_type

FROM

JSON\_TABLE,

LATERAL FLATTEN(input => parse\_json(data)) AS main\_node, -- Add comma before new LATERAL FLATTEN

LATERAL FLATTEN(input => main\_node.value:batter) AS batter,

LATERAL FLATTEN(input => main\_node.this:topping) AS topping;

SELECT \* FROM JSON\_TABLE;

**02/04/2025**

show stages;

list @S3\_STAGE;

CREATE OR REPLACE TABLE my\_table (

CI STRING,

DAYS STRING,

SERVICE\_TYPE STRING,

IN\_TIME STRING,

OUT\_TIME STRING,

CAREGIVER STRING

);

COPY INTO MY\_TABLE FROM @S3\_STAGE

FILE\_FORMAT = (TYPE = CSV FIELD\_OPTIONALLY\_ENCLOSED\_BY='"' SKIP\_HEADER=1);

SELECT \* FROM MY\_TABLE;

-- create a procedure to insert the data and schedule a task to insert the data every 5 minute.

CREATE OR REPLACE PROCEDURE CSV\_procedure()

RETURNS STRING

LANGUAGE SQL

AS

$$

BEGIN

COPY INTO MY\_TABLE

FROM @S3\_STAGE

FILE\_FORMAT = (TYPE = 'CSV' FIELD\_OPTIONALLY\_ENCLOSED\_BY='"' SKIP\_HEADER=1);

RETURN 'Data Inserted Successfully';

END;

$$;

call CSV\_procedure();

CREATE OR REPLACE TASK csv\_task

WAREHOUSE = COMPUTE\_WH

SCHEDULE = '5 MINUTE'

AS

CALL CSV\_procedure();

ALTER TASK csv\_task RESUME;

SELECT \* FROM MY\_TABLE;

ALTER TASK csv\_task SUSPEND;

SHOW TASKS IN SCHEMA ASSIGNMENT\_SCHEMA;

-- Prepare a time travel query for the table that you created to load the data which shows the data in the table when first loaded.

SELECT \* FROM MY\_TABLE AT (OFFSET=>-60\*5);