```
use role developer;
use database DEBADATTA_MOHANTY_DB;
create schema inventory_schema_debadatta;
use schema inventory_schema_debadatta;

create or replace TABLE DIM_DATE (
        DATE_SK NUMBER(38,0),
        DATE_ID VARCHAR(16),
        DATE DATE,
        MONTH_SEQ NUMBER(38,0),
        WEEK_SEQ NUMBER(38,0),
        QUARTER_SEQ NUMBER(38,0),
        YEAR NUMBER(38,0),
        DOW NUMBER(38,0),
        MOY NUMBER(38,0),
        DOM NUMBER(38,0),
        QOY NUMBER(38,0),
        FY_YEAR NUMBER(38,0),
        FY_QUARTER_SEQ NUMBER(38,0),
        FY_WEEK_SEQ NUMBER(38,0),
        DAY_NAME VARCHAR(9),
        QUARTER_NAME VARCHAR(6),
        HOLIDAY VARCHAR(1),
        WEEKEND VARCHAR(1),
        FOLLOWING_HOLIDAY VARCHAR(1),
        FIRST_DOM NUMBER(38,0),
        LAST_DOM NUMBER(38,0),
        SAME_DAY_LY NUMBER(38,0),
        SAME_DAY_LQ NUMBER(38,0),
```

```
	CURRENT_DAY VARCHAR(1),
	CURRENT_WEEK VARCHAR(1),
	CURRENT_MONTH VARCHAR(1),
	CURRENT_QUARTER VARCHAR(1),
	CURRENT_YEAR VARCHAR(1)
);

create or replace TABLE DIM_ITEM (
	ITEM_SK NUMBER(38,0),
	ITEM_ID VARCHAR(16),
	REC_START_DATE DATE,
	REC_END_DATE DATE,
	ITEM_DESC VARCHAR(200),
	CURRENT_PRICE NUMBER(7,2),
	WHOLESALE_COST NUMBER(7,2),
	BRAND_ID NUMBER(38,0),
	BRAND VARCHAR(50),
	CLASS_ID NUMBER(38,0),
	CLASS VARCHAR(50),
	CATEGORY_ID NUMBER(38,0),
	CATEGORY VARCHAR(50),
	MANUFACT_ID NUMBER(38,0),
	MANUFACT VARCHAR(50),
	SIZE VARCHAR(20),
	FORMULATION VARCHAR(20),
	COLOR VARCHAR(20),
	UNITS VARCHAR(10),
	CONTAINER VARCHAR(10),
	MANAGER_ID NUMBER(38,0),
```

```
	PRODUCT_NAME VARCHAR(50)
);

create or replace TABLE DIM_WAREHOUSE (
	WAREHOUSE_SK NUMBER(38,0),
	WAREHOUSE_ID VARCHAR(16),
	WAREHOUSE_NAME VARCHAR(20),
	WAREHOUSE_SQ_FT NUMBER(38,0),
	STREET_NUMBER VARCHAR(10),
	STREET_NAME VARCHAR(60),
	STREET_TYPE VARCHAR(15),
	SUITE_NUMBER VARCHAR(10),
	CITY VARCHAR(60),
	COUNTY VARCHAR(30),
	STATE VARCHAR(2),
	ZIP VARCHAR(10),
	COUNTRY VARCHAR(20),
	GMT_OFFSET NUMBER(5,2)
);

--CREATE TABLE DIM_DATE as
;

insert into DIM_DATE
select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."DATE_DIM";

--CREATE TABLE DIM_ITEM as
;
```

```
insert into DIM_ITEM
select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."ITEM";

--CREATE TABLE DIM_WAREHOUSE as
;

insert into DIM_WAREHOUSE
select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."WAREHOUSE";

create or replace TABLE F_INVENTORY (
        INV_DATE NUMBER(38,0),
        INV_ITEM NUMBER(38,0),
        INV_WAREHOUSE NUMBER(38,0),
        INV_QUANTITY_ON_HAND NUMBER(38,0)
);

alter table f_inventory  add column add_file_name varchar;
alter table f_inventory  add column source_add_file_row_number number;
alter table f_inventory  add column add_timestamp timestamp_NTZ;
alter table f_inventory  add column upd_file_name varchar;
alter table f_inventory  add column source_upd_file_row_number number;
alter table f_inventory  add column upd_timestamp timestamp_NTZ;

select * from DIM_DATE;

select * from DIM_ITEM;

select * from DIM_WAREHOUSE;
```

```
CREATE FILE FORMAT JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION = 'AUTO' ENABLE_OCTAL =
FALSE

ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE STRIP_NULL_VALUES = TRUE
IGNORE_UTF8_ERRORS = TRUE;


--create stage

create or replace stage JSON_INPUT_STAGE file_format = JSON_INPUT_FORMAT;


list @JSON_INPUT_STAGE;


--SELECT GET_DDL ('TABLE','DIM_DATE');




--- step-
1**********************************************************************************
*************

-- create a procedure to have working SQL TEXT

CREATE OR REPLACE PROCEDURE PROC_LOAD_DYNAMIC_PREP_05(TABLE_NAME STRING,STAGE_NAME
STRING)
   RETURNS STRING
   LANGUAGE JAVASCRIPT
   AS
   $$


   var result="";
         var stepnum="";
   // Build SQL Statements
         //SQL TEXT for FILE FORMAT
   var sql00 = "CREATE FILE FORMAT IF NOT EXISTS JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION
= 'AUTO' ENABLE_OCTAL = FALSE ";
```

```
   sql00 = sql00 + "ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE STRIP_NULL_VALUES =
FALSE IGNORE_UTF8_ERRORS = FALSE;";


          var stg_table = TABLE_NAME + "_STAGE";
    // create stage table


    var sql01 = "CREATE TABLE "+stg_table+" IF NOT EXISTS (COL1 VARIANT,FILENAME_JSON
STRING,ROW_NUMBER_JSON NUMBER,DTS_JSON TIMESTAMP_LTZ(9));";



   var sql02 = "COPY INTO "+ stg_table +" (col1,filename_json,row_number_json,dts_json) ";
                    sql02 = sql02 + " FROM (select $1,
METADATA$FILENAME,METADATA$FILE_ROW_NUMBER, current_timestamp from ";
                    sql02 = sql02 + " @"+ STAGE_NAME + "(file_format => JSON_INPUT_FORMAT))
ON_ERROR = SKIP_FILE PURGE = FALSE ;";


   result =  sql00 + sql01 + sql02;


   return result;
   $$;


call PROC_LOAD_DYNAMIC_PREP_05 ('F_INVENTORY','JSON_INPUT_STAGE/F_INVENTORY/JSON/');


CREATE FILE FORMAT IF NOT EXISTS JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION = 'AUTO'
ENABLE_OCTAL = FALSE ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE STRIP_NULL_VALUES
= FALSE IGNORE_UTF8_ERRORS = FALSE;


CREATE TABLE F_INVENTORY_STAGE IF NOT EXISTS (COL1 VARIANT,FILENAME_JSON
STRING,ROW_NUMBER_JSON NUMBER,DTS_JSON TIMESTAMP_LTZ(9));


COPY INTO F_INVENTORY_STAGE (col1,filename_json,row_number_json,dts_json)  FROM
```

```
(select $1, METADATA$FILENAME,METADATA$FILE_ROW_NUMBER, current_timestamp from
@JSON_INPUT_STAGE/F_INVENTORY/JSON/(file_format => JSON_INPUT_FORMAT))

ON_ERROR = SKIP_FILE PURGE = FALSE ;


select system$cancel_query('01a0f6c8-0c02-580f-0001-e9be00039c96');


select * from F_INVENTORY_STAGE;


--truncate table F_INVENTORY_STAGE;



--step-2 execute the SQL Text as SQL statement



CREATE OR REPLACE PROCEDURE PROC_LOAD_DYNAMIC_PREP_02 (TABLE_NAME
STRING,STAGE_NAME STRING)

   RETURNS STRING NOT NULL

   LANGUAGE JAVASCRIPT

   AS

   $$


   var result="";

         var stepnum="";

   // Build SQL Statements

         //SQL Statement for FILE FORMAT

   var sql00 = "CREATE FILE FORMAT IF NOT EXISTS JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION
= 'AUTO' ENABLE_OCTAL = FALSE ";

   sql00 = sql00 + "ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE STRIP_NULL_VALUES =
TRUE IGNORE_UTF8_ERRORS = TRUE;";
```

```
        var stg_table = TABLE_NAME+"_STAGE";
    // create stage table

    var sql01 = "CREATE TABLE "+stg_table+" IF NOT EXISTS (COL1 VARIANT,FILENAME_JSON
STRING,ROW_NUMBER_JSON NUMBER,DTS_JSON TIMESTAMP_LTZ(9));";


   var sql02 = "COPY INTO "+ stg_table +" (col1,filename_json,row_number_json,dts_json) ";
                    sql02 = sql02 + " FROM (select $1,
METADATA$FILENAME,METADATA$FILE_ROW_NUMBER, current_timestamp from ";
                    sql02 = sql02 + " @"+ STAGE_NAME + "(file_format => JSON_INPUT_FORMAT))
ON_ERROR = SKIP_FILE PURGE = FALSE ;";


  //Execute SQL
   try {
   //*****************Create SQL Statement ***************
   stepnum = 1;
      var stmt00 = snowflake.createStatement( { sqlText: sql00 } );
      stmt00.execute();
   //**************ends one statement execute************************
      stepnum =2;
      var stmt01 = snowflake.createStatement( { sqlText: sql01 } );
      stmt01.execute();
         stepnum = 3;
       var stmt02 = snowflake.createStatement( { sqlText: sql02 } );
      stmt02.execute();
      result = "Copy Command Succeeded!";
       return result;
   }
```

```
   catch (err)  {
      result =  "Failed in Step:"+stepnum + ": Code: " + err.code + "\n  State: " + err.state;
      result += "\n  Message: " + err.message;
      result += "\nStack Trace:\n" + err.stackTraceTxt;
      return result;
   }
    $$;

call PROC_LOAD_DYNAMIC_PREP_02 ('F_INVENTORY','JSON_INPUT_STAGE/F_INVENTORY/JSON/');

--final SP

CREATE OR REPLACE PROCEDURE PROC_LOAD_DYNAMIC(TABLE_NAME STRING,STAGE_NAME STRING)
   RETURNS STRING
   LANGUAGE JAVASCRIPT
   EXECUTE AS CALLER -- Caller prevs...
   AS
   $$

   var result="";
         var stepnum="";
   // Build SQL Statements
         //SQL Statement for FILE FORMAT
   var sql00 = "CREATE FILE FORMAT IF NOT EXISTS JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION
= 'AUTO' ENABLE_OCTAL = FALSE  ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE
STRIP_NULL_VALUES = FALSE IGNORE_UTF8_ERRORS = FALSE;";
         var stg_table = TABLE_NAME+"_JSON_STAGE";
   var stg_view = TABLE_NAME+"_JS_VIEW";
```

```
// create table

var sql01 = "CREATE  TEMPORARY TABLE "+stg_table+" IF NOT EXISTS (COL1
VARIANT,FILENAME_JSON STRING,ROW_NUMBER_JSON NUMBER,DTS_JSON TIMESTAMP_LTZ(9));";

//Build your COPY COMMAND

var sql02 = "COPY INTO "+ stg_table +" (col1,filename_json,row_number_json,dts_json) ";

        sql02 = sql02 + " FROM (select $1,
METADATA$FILENAME,METADATA$FILE_ROW_NUMBER, current_timestamp from ";

        sql02 = sql02 + " @"+ STAGE_NAME + "(file_format => JSON_INPUT_FORMAT))
ON_ERROR = SKIP_FILE PURGE = FALSE ;";

    //Build View on the JSON Data

var sql03 = "CREATE OR REPLACE VIEW  "+stg_view+" AS SELECT $1:INVENTORY_DATE::STRING AS
INVENTORY_DATE,$1:INV_QUANTITY_ON_HAND::NUMBER AS STOCK_IN_HAND";

        sql03+= ",$1:ITEM_DET.ITEM_ID::STRING AS
ITEM_ID,$1:WAREHOUSE_DET.warehouse_id::STRING AS WAREHOUSE_ID,FILENAME_JSON AS
INPUT_FILE_NAME";

        sql03+= ",ROW_NUMBER_JSON AS ROW_NUM_IN_FILE,DTS_JSON AS LOAD_DATE
FROM "+stg_table+";";

//Build the merge statement

var sql04 = " merge into "+ TABLE_NAME+ " fi using (select dd.date_sk,
dw.WAREHOUSE_SK,di.ITEM_SK,iv.STOCK_IN_HAND, iv.input_file_name ,iv.ROW_NUM_IN_FILE";

  sql04+= " from DIM_WAREHOUSE dw,DIM_ITEM di,DIM_DATE dd, "+stg_view+" iv where dd.DATE =
to_date(iv.INVENTORY_DATE,'MM/DD/YYYY') and dw.WAREHOUSE_ID = iv.WAREHOUSE_ID";

  sql04+= " and di.ITEM_ID = iv.ITEM_ID and di.REC_END_DATE is null) ivw on fi.INV_DATE =
ivw.date_sk and fi.INV_ITEM = ivw.ITEM_SK and fi.INV_WAREHOUSE =ivw.WAREHOUSE_SK";
```

```
    sql04+= " when matched then update set upd_file_name = ivw.input_file_name ,
source_upd_file_row_number = ivw.ROW_NUM_IN_FILE , upd_timestamp = current_timestamp,
INV_QUANTITY_ON_HAND =ivw.STOCK_IN_HAND";

   sql04+= " when not matched then insert
(INV_DATE,INV_ITEM,INV_WAREHOUSE,INV_QUANTITY_ON_HAND,add_file_name,source_add_file_ro
w_number,add_timestamp)";

   sql04+= " values (
ivw.date_sk,ivw.ITEM_SK,ivw.WAREHOUSE_SK,ivw.STOCK_IN_HAND,ivw.input_file_name,ivw.ROW_NU
M_IN_FILE,current_timestamp );";


 var  sql05 = "SELECT * FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()));";


    //Execute SQL
   try {
   //*****************execute block***************
      var stmt00 = snowflake.createStatement( { sqlText: sql00 } );
      stmt00.execute();
   //**************ends ************************
      var stmt01 = snowflake.createStatement( { sqlText: sql01 } );
      stmt01.execute();
       var stmt02 = snowflake.createStatement( { sqlText: sql02 } );
      stmt02.execute();
      var stmt03 = snowflake.createStatement( { sqlText: sql03 } );
      stmt03.execute();
      var stmt04 = snowflake.createStatement( { sqlText: sql04 } );
      stmt04.execute();
      var stmt05 = snowflake.createStatement( { sqlText: sql05 } );
      rs01 = stmt05.execute();
      rs01.next();
      result = " Succeeded! Rows inserted: " + rs01.getColumnValue(1) + ", Rows updated: " +
rs01.getColumnValue(2);
```

```
        return result;
    }
    catch (err)  {
        result =  "Failed: Code: " + err.code + "\n  State: " + err.state;
        result += "\n  Message: " + err.message;
        result += "\nStack Trace:\n" + err.stackTraceTxt;
        return result;
    }

    $$;

truncate table F_INVENTORY;

select count(*) from F_INVENTORY;

select count(*) from F_INVENTORY_STAGE;

call PROC_LOAD_DYNAMIC ('F_INVENTORY','JSON_INPUT_STAGE/F_INVENTORY/JSON/');

list @JSON_INPUT_STAGE/F_INVENTORY/JSON/;

select * from F_INVENTORY_JS_VIEW;

select * from F_INVENTORY;

truncate table f_inventory;

Failed: Code: 100183
  State: P0000
```

Message: SQL compilation error: error line 1 at position 683

invalid identifier 'INV_DATE'

Stack Trace:

At Statement.execute, line 50 position 15;

```
CREATE OR REPLACE PROCEDURE PROC_LOAD_DYNAMIC_BKP(TABLE_NAME STRING,STAGE_NAME
STRING)
   RETURNS STRING
   LANGUAGE JAVASCRIPT
   EXECUTE AS CALLER -- Caller prevs...
   AS
   $$

   var result="";
         var stepnum="";
   // Build SQL Statements
         //SQL Statement for FILE FORMAT
   var sql00 = "CREATE FILE FORMAT IF NOT EXISTS JSON_INPUT_FORMAT TYPE = 'JSON' COMPRESSION
= 'AUTO' ENABLE_OCTAL = FALSE  ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = TRUE
STRIP_NULL_VALUES = FALSE IGNORE_UTF8_ERRORS = FALSE;";
         var stg_table = TABLE_NAME+"_JSON_STAGE";
   var stg_view = TABLE_NAME+"_JS_VIEW";
    // create table

    var sql01 = "CREATE  TEMPORARY TABLE "+stg_table+" IF NOT EXISTS (COL1
VARIANT,FILENAME_JSON STRING,ROW_NUMBER_JSON NUMBER,DTS_JSON TIMESTAMP_LTZ(9));";
   //Build your COPY COMMAND


   var sql02 = "COPY INTO "+ stg_table +" (col1,filename_json,row_number_json,dts_json) ";
```

```
                sql02 = sql02 + " FROM (select $1,
METADATA$FILENAME,METADATA$FILE_ROW_NUMBER, current_timestamp from ";

                sql02 = sql02 + " @"+ STAGE_NAME + "(file_format => JSON_INPUT_FORMAT))
ON_ERROR = SKIP_FILE PURGE = FALSE ;";


        //Build View on the JSON Data


  var sql03 = "CREATE OR REPLACE VIEW  "+stg_view+" AS SELECT $1:INVENTORY_DATE::STRING AS
INVENTORY_DATE,$1:INV_QUANTITY_ON_HAND::NUMBER AS STOCK_IN_HAND";

                sql03+= ",$1:ITEM_DET.ITEM_ID::STRING AS
ITEM_ID,$1:WAREHOUSE_DET.warehouse_id::STRING AS WAREHOUSE_ID,FILENAME_JSON AS
INPUT_FILE_NAME";

                sql03+= ",ROW_NUMBER_JSON AS ROW_NUM_IN_FILE,DTS_JSON AS LOAD_DATE
FROM "+stg_table+";";


  //Build the merge statement


 var sql04 = " merge into "+ TABLE_NAME+ " fi using (select dd.date_sk,
dw.WAREHOUSE_SK,di.ITEM_SK,iv.STOCK_IN_HAND, iv.input_file_name ,iv.ROW_NUM_IN_FILE";

    sql04+= " from DIM_WAREHOUSE dw,DIM_ITEM di,DIM_DATE dd, "+stg_view+" iv where dd.DATE =
to_date(iv.INVENTORY_DATE,'MM/DD/YYYY') and dw.WAREHOUSE_ID = iv.WAREHOUSE_ID";

    sql04+= " and di.ITEM_ID = iv.ITEM_ID and di.REC_END_DATE is null) ivw on fi.INV_DATE =
ivw.date_sk and fi.INV_ITEM = ivw.ITEM_SK and fi.INV_WAREHOUSE =ivw.WAREHOUSE_SK";

    sql04+= " when matched then update set upd_file_name = ivw.input_file_name ,
source_upd_file_row_number = ivw.ROW_NUM_IN_FILE , upd_timestamp = current_timestamp,
INV_QUANTITY_ON_HAND =ivw.STOCK_IN_HAND";

   sql04+= " when not matched then insert
(INV_DATE,INV_ITEM,INV_WAREHOUSE,INV_QUANTITY_ON_HAND,add_file_name,source_add_file_ro
w_number,add_timestamp)";

   sql04+= " values (
ivw.date_sk,ivw.ITEM_SK,ivw.WAREHOUSE_SK,ivw.STOCK_IN_HAND,ivw.input_file_name,ivw.ROW_NU
M_IN_FILE,current_timestamp );";

 return sql04;
```

```
   $$;

call PROC_LOAD_DYNAMIC_BKP ('F_INVENTORY','JSON_INPUT_STAGE/F_INVENTORY/JSON/');

select get_ddl('table','F_INVENTORY');



alter table f_inventory  drop column upd_date;

select get_ddl('table','f_inventory');

 merge into F_INVENTORY fi using
 (select dd.date_sk, dw.WAREHOUSE_SK,di.ITEM_SK,iv.STOCK_IN_HAND, iv.input_file_name
,iv.ROW_NUM_IN_FILE
  from DIM_WAREHOUSE dw,DIM_ITEM di,DIM_DATE dd, F_INVENTORY_JS_VIEW iv
  where dd.DATE = to_date(iv.INVENTORY_DATE,'MM/DD/YYYY') and dw.WAREHOUSE_ID =
iv.WAREHOUSE_ID
  and di.ITEM_ID = iv.ITEM_ID and di.REC_END_DATE is null) ivw
  on fi.INV_DATE = ivw.date_sk and fi.INV_ITEM = ivw.ITEM_SK
  and fi.INV_WAREHOUSE =ivw.WAREHOUSE_SK
  when matched then
  update set upd_file_name = ivw.input_file_name , source_upd_file_row_number =
ivw.ROW_NUM_IN_FILE ,
  upd_timestamp = current_timestamp, INV_QUANTITY_ON_HAND =ivw.STOCK_IN_HAND
  when not matched then insert
(INV_DATE,INV_ITEM,INV_WAREHOUSE,INV_QUANTITY_ON_HAND,add_file_name,
                        source_add_file_row_number,add_timestamp) values
 (
ivw.date_sk,ivw.ITEM_SK,ivw.WAREHOUSE_SK,ivw.STOCK_IN_HAND,ivw.input_file_name,ivw.ROW_NU
M_IN_FILE,current_timestamp );
```