**BUSINESS METADATA TABLE DOCUMENT**

1. In Snowflake, we have created a Business Metadata Table which lies on the Governance Database.
2. The main objective of this business metadata table is that, it contains all the table metadata like column names, column datatypes, source\_system, etc
3. This table will be provided to the business to update the Business Term, Description, Data Owner, Data Steward and Approval Status
4. Once these columns have been filled in, this Business Metadata table serves as one glossary for all the tables in Snowflake

CREATE OR REPLACE PROCEDURE DATADICTIONARY\_DEV.DEV\_DICTIONARY.UPDATE\_METADATA\_PROC()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS

$$

DECLARE

cursor1 CURSOR FOR select \* from RAW\_DEV.information\_schema.tables where table\_schema = 'DEV\_RAW\_SALESFORCE';

result\_record RESULTSET;

db\_name string;

schema\_name string;

table\_name string;

column\_name string;

data\_type string;

numeric\_precision number;

numeric\_scale number;

character\_maximum\_length number;

tag\_name string;

tag\_value string;

source\_system string;

BEGIN

OPEN cursor1;

FOR record IN cursor1 DO

db\_name := record.TABLE\_CATALOG;

schema\_name := record.TABLE\_SCHEMA;

table\_name := record.TABLE\_NAME;

LET domain := 'COLUMN';

result\_record := (SELECT COLUMN\_NAME,DATA\_TYPE,NUMERIC\_PRECISION,NUMERIC\_SCALE,CHARACTER\_MAXIMUM\_LENGTH from RAW\_DEV.information\_schema.columns where table\_name = :table\_name);

FOR column\_record IN result\_record DO

column\_name := column\_record.COLUMN\_NAME;

data\_type := column\_record.DATA\_TYPE;

numeric\_precision := column\_record.NUMERIC\_PRECISION;

numeric\_scale := column\_record.NUMERIC\_SCALE;

character\_maximum\_length := column\_record.CHARACTER\_MAXIMUM\_LENGTH;

SELECT TAG\_NAME, TAG\_VALUE INTO tag\_name, tag\_value FROM SNOWFLAKE.ACCOUNT\_USAGE.TAG\_REFERENCES WHERE object\_database = :db\_name AND object\_schema = :schema\_name AND object\_name = :table\_name AND COLUMN\_NAME = :column\_name AND domain = :domain;

INSERT INTO DATADICTIONARY\_DEV.DEV\_DICTIONARY.METADATA\_STAGING\_TABLE(

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

COLUMN\_NAME,

COLUMN\_DATATYPE,

SOURCE\_SYSTEM,

TAGS,

CREATED\_DATETIME

)

VALUES(

:db\_name,

:schema\_name,

:table\_name,

:column\_name,

CASE

WHEN :data\_type = 'NUMBER' THEN :data\_type || '(' || :numeric\_precision || ',' || :numeric\_scale || ')'

WHEN :data\_type = 'TEXT' THEN 'VARCHAR(' || :character\_maximum\_length || ')'

WHEN :data\_type IN ('DATE','BOOLEAN','TIMESTAMP\_NTZ') THEN :data\_type

END,

CASE

WHEN :schema\_name = 'DEV\_RAW\_SALESFORCE' THEN 'SALESFORCE'

WHEN :schema\_name = 'DEV\_RAW\_MICROSOFTGRAPH' THEN 'MICROSOFT GRAPH'

WHEN :schema\_name = 'DEV\_RAW\_MERAKI' THEN 'MERAKI'

END,

CASE

WHEN :tag\_name IS NOT NULL AND :tag\_value IS NOT NULL THEN :tag\_name || ' - ' || :tag\_value

END,

CURRENT\_TIMESTAMP()

);

END FOR;

END FOR;

-- Merge the staging table into the main metadata table

MERGE INTO DATADICTIONARY\_DEV.DEV\_DICTIONARY.BUSINESS\_METADATA\_TABLE\_TEST AS target

USING DATADICTIONARY\_DEV.DEV\_DICTIONARY.METADATA\_STAGING\_TABLE AS source

ON target.DATABASE\_NAME = source.DATABASE\_NAME

AND target.SCHEMA\_NAME = source.SCHEMA\_NAME

AND target.TABLE\_NAME = source.TABLE\_NAME

AND target.COLUMN\_NAME = source.COLUMN\_NAME

WHEN MATCHED THEN

UPDATE SET

COLUMN\_DATATYPE = source.COLUMN\_DATATYPE,

SOURCE\_SYSTEM = source.SOURCE\_SYSTEM,

TAGS = source.TAGS,

CREATED\_DATETIME = source.CREATED\_DATETIME

WHEN NOT MATCHED THEN

INSERT (

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

COLUMN\_NAME,

COLUMN\_DATATYPE,

SOURCE\_SYSTEM,

TAGS,

CREATED\_DATETIME

)

VALUES (

source.DATABASE\_NAME,

source.SCHEMA\_NAME,

source.TABLE\_NAME,

source.COLUMN\_NAME,

source.COLUMN\_DATATYPE,

source.SOURCE\_SYSTEM,

source.TAGS,

source.CREATED\_DATETIME

);

-- Clear the staging table

TRUNCATE TABLE DATADICTIONARY\_DEV.DEV\_DICTIONARY.METADATA\_STAGING\_TABLE;

CLOSE cursor1;

RETURN 'Procedure Completed Successfully';

END;

$$;

