**STORED PROCEDURES DOCUMENT**

**Load Historical Data from Stage to RAW tables:**

CREATE OR REPLACE PROCEDURE RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.MICROSOFT\_HISTORICAL\_LOAD()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE RecordCount\_Access NUMBER;

RecordCount\_AccessEmail NUMBER;

RecordCount\_Devices NUMBER;

RecordCount\_Email NUMBER;

RecordCount\_Employee NUMBER;

RecordCount\_Intune NUMBER;

RecordCount\_Location NUMBER;

RecordCount\_Visits NUMBER;

TableName VARCHAR;

BEGIN

-- Author: Bharath Sridhar

-- Created: 02/04/2025

-- Description: Historical load Procedure

-- Modifications: 02/04/2025: Created the procedure for Microsoft Historical Load

CREATE TABLE IF NOT EXISTS HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE VARCHAR(16777216),

DATABASE\_NAME VARCHAR(16777216),

SCHEMA\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

ROW\_COUNT NUMBER(38,0),

LOAD\_TIMESTAMP TIMESTAMP\_NTZ(9),

PROCEDURE\_NAME VARCHAR(16777216),

STATUS VARCHAR(16777216),

ERROR\_MESSAGE VARCHAR(16777216)

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/access\_cards.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''ACCESS\_CARDS'';

SELECT "rows\_loaded" INTO RecordCount\_Access FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''ACCESS\_CARDS'',

:RecordCount\_Access,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS\_EMAIL\_MAP FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/access\_cards\_email\_map.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''ACCESS\_CARDS\_EMAIL\_MAP'';

SELECT "rows\_loaded" INTO RecordCount\_AccessEmail FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''ACCESS\_CARDS\_EMAIL\_MAP'',

:RecordCount\_AccessEmail,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.DEVICES FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/devices.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''DEVICES'';

SELECT "rows\_loaded" INTO RecordCount\_Devices FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''DEVICES'',

:RecordCount\_Devices,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.EMAIL\_ACTIVITY FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/email\_activity.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''EMAIL\_ACTIVITY'';

SELECT "rows\_loaded" INTO RecordCount\_Email FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''EMAIL\_ACTIVITY'',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.EMPLOYEES FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/employees.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''EMPLOYEES'';

SELECT "rows\_loaded" INTO RecordCount\_Employee FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''EMPLOYEES'',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.INTUNE\_DATA FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/intune\_data.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''INTUNE\_DATA'';

SELECT "rows\_loaded" INTO RecordCount\_Intune FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''INTUNE\_DATA'',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.LOCATIONS FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/location.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''LOCATIONS'';

SELECT "rows\_loaded" INTO RecordCount\_Location FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''LOCATIONS'',

:RecordCount\_Location,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

COPY INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.VISITS FROM ''@"RAW\_DEV"."DEV\_RAW\_MICROSOFTGRAPH"."MICROSOFT\_STAGE"/visits.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

TableName := ''VISITS'';

SELECT "rows\_loaded" INTO RecordCount\_Visits FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

''VISITS'',

:RecordCount\_Visits,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''HARMONIZED\_DEV'',

''DEV\_HAR\_MICROSOFTGRAPH'',

:TableName,

0,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_PROC'',

''FAILURE'',

:SQLERRM

);

END;

';

**Load Historical Data from RAW to Harmonized:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.MICROSOFT\_HISTORICAL\_LOAD()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS

$$

DECLARE RecordCount\_Access NUMBER;

RecordCount\_AccessEmail NUMBER;

RecordCount\_Devices NUMBER;

RecordCount\_Email NUMBER;

RecordCount\_Employee NUMBER;

RecordCount\_Intune NUMBER;

RecordCount\_Location NUMBER;

RecordCount\_Visits NUMBER;

TableName STRING;

BEGIN

-- Author: Bharath Sridhar

-- Created: 02/04/2025

-- Description: Historical load Procedure

-- Modifications: 02/04/2025: Created the procedure for Microsoft Historical Load to Harmonized DB

CREATE TABLE IF NOT EXISTS HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE VARCHAR(16777216),

DATABASE\_NAME VARCHAR(16777216),

SCHEMA\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

ROW\_COUNT NUMBER(38,0),

LOAD\_TIMESTAMP TIMESTAMP\_NTZ(9),

PROCEDURE\_NAME VARCHAR(16777216),

STATUS VARCHAR(16777216),

ERROR\_MESSAGE VARCHAR(16777216)

);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.ACCESS\_CARDS (

ID,

USERNAME,

VISIT\_DATE,

LOCATION,

EMAIL

)

SELECT

ID,

USERNAME,

VISIT\_DATE,

LOCATION,

EMAIL

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS;

TableName := 'ACCESS\_CARDS';

RecordCount\_Access := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'ACCESS\_CARDS',

:RecordCount\_Access,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL

);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.ACCESS\_CARDS\_EMAIL\_MAP(

INDEX,

USERNAME,

EMAIL

)

SELECT

INDEX,

USERNAME,

EMAIL

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS\_EMAIL\_MAP;

TableName := 'ACCESS\_CARDS\_EMAIL\_MAP';

RecordCount\_AccessEmail := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'ACCESS\_CARDS\_EMAIL\_MAP',

:RecordCount\_AccessEmail,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL

);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.DEVICES(

DEVICEID,

OPERATINGSYSTEM,

MACADDRESS,

EMPLOYEEEMAIL

)

SELECT

DEVICEID,

OPERATINGSYSTEM,

MACADDRESS,

EMPLOYEEEMAIL

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.DEVICES;

TableName := 'DEVICES';

RecordCount\_Devices := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'DEVICES',

:RecordCount\_Devices,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.EMAIL\_ACTIVITY(

EMAILACTIVITYID,

OBJECTID,

EMAIL,

REPORTREFRESHDATE,

DISPLAYNAME,

SENDCOUNT,

RECEIVECOUNT,

MEETINGCREATEDCOUNT,

MEETINGINTERACTEDCOUNT,

REPORTPERIOD,

DATASOURCE,

ISDELETED,

DELETEDDATE,

LASTACTIVITYDATE,

ASSIGNEDPRODUCTS,

READCOUNT

)

SELECT

EMAILACTIVITYID,

OBJECTID,

EMAIL,

REPORTREFRESHDATE,

DISPLAYNAME,

SENDCOUNT,

RECEIVECOUNT,

MEETINGCREATEDCOUNT,

MEETINGINTERACTEDCOUNT,

REPORTPERIOD,

DATASOURCE,

NULL,

NULL,

NULL,

NULL,

0

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.EMAIL\_ACTIVITY;

TableName := 'EMAIL\_ACTIVITY';

RecordCount\_Email := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'EMAIL\_ACTIVITY',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.EMPLOYEES(

OBJECTID,

FIRSTNAME,

LASTNAME,

DISPLAYNAME,

EMAIL,

DEPARTMENT,

OFFICELOCATION,

EMPLOYEETYPE,

HIREDATE,

ACCOUNTENABLED

)

SELECT

OBJECTID,

FIRSTNAME,

LASTNAME,

DISPLAYNAME,

EMAIL,

DEPARTMENT,

OFFICELOCATION,

EMPLOYEETYPE,

HIREDATE,

ACCOUNTENABLED

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.EMPLOYEES;

TableName := 'EMPLOYEES';

RecordCount\_Employee := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'EMPLOYEES',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.INTUNE\_DATA(

ATTENDANCEDATE,

ACTIVATIONLOCKBYPASSCODE,

ANDROIDSECURITYPATCHLEVEL,

AZUREADDEVICEID,

AZUREADREGISTERED,

COMPLIANCEGRACEPERIODEXPIRATIONDATETIME,

COMPLIANCESTATE,

CONFIGURATIONMANAGERCLIENTENABLEDFEATURES,

DEVICEACTIONRESULTS,

DEVICECATEGORYDISPLAYNAME,

DEVICEENROLLMENTTYPE,

DEVICEHEALTHATTESTATIONSTATE,

DEVICENAME,

DEVICEREGISTRATIONSTATE,

EASACTIVATED,

EASACTIVATIONDATETIME,

EASDEVICEID,

EMAILADDRESS,

ENROLLEDDATETIME,

ENROLLMENTPROFILENAME,

ETHERNETMACADDRESS,

EXCHANGEACCESSSTATE,

EXCHANGEACCESSSTATEREASON,

EXCHANGELASTSUCCESSFULSYNCDATETIME,

FREESTORAGESPACEINBYTES,

ICCID,

ID,

IMEI,

ISENCRYPTED,

ISSUPERVISED,

JAILBROKEN,

LASTSYNCDATETIME,

MANAGEDDEVICENAME,

MANAGEDDEVICEOWNERTYPE,

MANAGEMENTAGENT,

MANAGEMENTCERTIFICATEEXPIRATIONDATE,

MANUFACTURER,

MEID,

MODEL,

NOTES,

OPERATINGSYSTEM,

OSVERSION,

PARTNERREPORTEDTHREATSTATE,

PHONENUMBER,

PHYSICALMEMORYINBYTES,

REMOTEASSISTANCESESSIONERRORDETAILS,

REMOTEASSISTANCESESSIONURL,

REQUIREUSERENROLLMENTAPPROVAL,

SERIALNUMBER,

SUBSCRIBERCARRIER,

TOTALSTORAGESPACEINBYTES,

UDID,

USERDISPLAYNAME,

USERID,

USERPRINCIPALNAME,

WIFIMACADDRESS

)

SELECT

ATTENDANCEDATE,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

DEVICENAME,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

OPERATINGSYSTEM,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

USERDISPLAYNAME,

NULL,

NULL,

WIFIMACADDRESS

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.INTUNE\_DATA;

TableName := 'INTUNE\_DATA';

RecordCount\_Intune := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'INTUNE\_DATA',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.LOCATIONS(

LOCATIONID,

LOCATIONNAME

)

SELECT

LOCATIONID,

LOCATIONNAME

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.LOCATIONS;

TableName := 'LOCATIONS';

RecordCount\_Location := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'LOCATIONS',

:RecordCount\_Location,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.VISITS(

VISITID,

EMPLOYEEEMAIL,

LOCATIONNAME,

VISITDATETIME,

VISITDATE,

MACADDRESS

)

SELECT

VISITID,

EMPLOYEEEMAIL,

LOCATIONNAME,

VISITDATETIME,

VISITDATE,

MACADDRESS

FROM RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.VISITS;

TableName := 'VISITS';

RecordCount\_Visits := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'VISITS',

:RecordCount\_Visits,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'SUCCESS',

NULL);

EXCEPTION

WHEN OTHER THEN

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

:TableName,

0,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_HISTORICAL\_LOAD\_PROC',

'FAILURE',

:sqlerrm);

END;

$$;

**Setup Connection for Microsoft Graph API:**

1. Create Network Rule using the required URLs using SECURITYADMIN role

CREATE OR REPLACE NETWORK RULE Graph\_Rule

MODE = EGRESS

TYPE = HOST\_PORT

VALUE\_LIST = ('graph.microsoft.com','login.microsoftonline.com','reportsncu.office.com');

1. Create External Access Integration using the above created Network Rule using ACCOUNTADMIN role

CREATE OR REPLACE EXTERNAL ACCESS INTEGRATION Graph\_API\_Access

ALLOWED\_NETWORK\_RULES = (Graph\_Rule)

ENABLED = TRUE;

1. Provide access to the fr\_dev\_dataengineer role to use this External Access Integration

GRANT USAGE ON INTEGRATION Graph\_API\_Access TO ROLE fr\_dev\_dataengineer;

**Load Continuous data from API to RAW layer:**

CREATE OR REPLACE PROCEDURE RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.MICROSOFT\_GRAPHAPI\_CONTINUOUSLOAD()

RETURNS VARCHAR

LANGUAGE PYTHON

RUNTIME\_VERSION = '3.10'

PACKAGES = ('snowflake-snowpark-python','requests')

HANDLER = 'main'

EXTERNAL\_ACCESS\_INTEGRATIONS = (GRAPH\_API\_ACCESS)

EXECUTE AS OWNER

AS '

import \_snowflake

import requests

import snowflake.connector

import json

import pandas as pd

from io import StringIO

from requests.auth import HTTPBasicAuth

from snowflake.snowpark import Session

def main(session: Session):

#Step 1: Get Access Token for MS Graph

# Define your Azure AD and Snowflake credentials here

client\_id = ''2de23d3b-06e8-4ab5-b721-392718926a5e''

client\_secret = ''s008Q~EHidWlf3jkLvEGRY4PM~hOeKtTeKB15boD''

tenant\_id = ''e1772763-63bf-431a-837c-6301e6b62d6a''

oauth\_url = f"https://login.microsoftonline.com/{tenant\_id}/oauth2/v2.0/token"

# Define the payload for OAuth token request

payload = {

''grant\_type'': ''client\_credentials'',

''client\_id'': client\_id,

''client\_secret'': client\_secret,

''scope'': ''https://graph.microsoft.com/.default'' # Scopes you want access to

}

# Make the request to get the OAuth token

response = requests.post(oauth\_url, data=payload)

# If the request was successful, return the access token

if response.status\_code == 200:

token = response.json()[''access\_token'']

else:

raise Exception(f"Error fetching token: {response.text}")

# Step 2: Call Microsoft Graph API to get Intune Data

api\_url\_intune = f"https://graph.microsoft.com/v1.0/deviceManagement/managedDevices"

api\_url\_employee = f"https://graph.microsoft.com/v1.0/users?$select=ID,givenName,surname,displayName,userPrincipalName,department,employeeHireDate,employeeType,Accountenabled,officeLocation"

api\_url\_email = f"https://graph.microsoft.com/v1.0/reports/getEmailActivityUserDetail(period=''D7'')"

# Define headers for the API request, including the OAuth token for authentication

headers = {

''Authorization'': f''Bearer {token}'',

''Content-Type'': ''application/json''

}

# Ensure raw table exists

session.sql("CREATE TABLE IF NOT EXISTS RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA (data VARIANT)").collect()

session.sql("CREATE TABLE IF NOT EXISTS RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_EMPLOYEE\_DATA (data VARIANT)").collect()

session.sql("CREATE TABLE IF NOT EXISTS RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_EMAIL\_DATA (data VARIANT)").collect()

# Ensure audit table exists

session.sql("""

CREATE TABLE IF NOT EXISTS HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE VARCHAR(16777216),

DATABASE\_NAME VARCHAR(16777216),

SCHEMA\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

ROW\_COUNT NUMBER(38,0),

LOAD\_TIMESTAMP TIMESTAMP\_NTZ(9),

PROCEDURE\_NAME VARCHAR(16777216),

STATUS VARCHAR(16777216),

ERROR\_MESSAGE VARCHAR(16777216)

);

""").collect()

################ Intune Data #######################

try:

response\_intune = requests.get(api\_url\_intune, headers=headers)

if response\_intune.status\_code == 200:

graph\_data\_intune = response\_intune.json()

# Insert data into raw table

result\_intune = session.sql("INSERT INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA (data) SELECT PARSE\_JSON(?)", [json.dumps(graph\_data\_intune)]).collect()

row\_count\_intune = len(graph\_data\_intune) if isinstance(graph\_data\_intune, list) else 1 # Adjust count for lists

# Insert audit log (Success)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_INTUNE\_DATA'',?, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''SUCCESS'', NULL)

""", [row\_count\_intune]).collect()

else:

raise Exception(f"Error calling Microsoft Graph API: {response\_intune.text}")

except Exception as e:

error\_message = str(e)

# Insert audit log (Failure)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_INTUNE\_DATA'',NULL, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''FAILURE'', ?)

""", [error\_message]).collect()

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_DEV.DEV\_RAW\_MERAKI.EMAIL WHERE FLAG=TRUE;"

email\_result = session.sql(email\_query).collect()

email\_list = [row["EMAIL\_ID"] for row in email\_result]

if email\_list:

email\_addresses = ",".join(email\_list)

# Send email notification

session.sql("""

CALL SYSTEM$SEND\_EMAIL(

''email\_integration'',

?,

?,

?

)

""", [email\_addresses, f"{location} Microsoft Graph API Failed", error\_message]).collect()

return f"Procedure failed: {error\_message}"

################ Employee Data #######################

try:

response\_employee = requests.get(api\_url\_employee, headers=headers)

if response\_employee.status\_code == 200:

graph\_data\_employee = response\_employee.json()

# Insert data into raw table

result\_employee = session.sql("INSERT INTO RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_EMPLOYEE\_DATA (data) SELECT PARSE\_JSON(?)", [json.dumps(graph\_data\_employee)]).collect()

row\_count\_employee = len(graph\_data\_employee) if isinstance(graph\_data\_employee, list) else 1 # Adjust count for lists

# Insert audit log (Success)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_EMPLOYEE\_DATA'',?, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''SUCCESS'', NULL)

""", [row\_count\_employee]).collect()

else:

raise Exception(f"Error calling Microsoft Graph API: {response\_employee.text}")

except Exception as e:

error\_message = str(e)

# Insert audit log (Failure)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_EMPLOYEE\_DATA'', NULL , CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''SUCCESS'', ?)

""", [error\_message]).collect()

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_DEV.DEV\_RAW\_MERAKI.EMAIL WHERE FLAG=TRUE;"

email\_result = session.sql(email\_query).collect()

email\_list = [row["EMAIL\_ID"] for row in email\_result]

if email\_list:

email\_addresses = ",".join(email\_list)

# Send email notification

session.sql("""

CALL SYSTEM$SEND\_EMAIL(

''email\_integration'',

?,

?,

?

)

""", [email\_addresses, f"{location} Microsoft Graph API Failed", error\_message]).collect()

return f"Procedure failed: {error\_message}"

################ Email Data #######################

try:

response\_email = requests.get(api\_url\_email, headers=headers)

if response\_email.status\_code == 200:

graph\_data\_email = StringIO(response\_email.text)

graph\_data\_df = pd.read\_csv(graph\_data\_email)

graph\_data\_df.rename(columns = {''Report Refresh Date'':''REPORTREFRESHDATE'',''User Principal Name'':''USERPRINCIPALNAME'',''Display Name'':''DISPLAYNAME'',''Is Deleted'':''ISDELETED'',''Deleted Date'':''DELETEDDATE'',''Last Activity Date'':''LASTACTIVITYDATE'',''Send Count'':''SENDCOUNT'',''Receive Count'':''RECEIVECOUNT'',''Read Count'':''READCOUNT'',''Meeting Created Count'':''MEETINGCREATEDCOUNT'',''Meeting Interacted Count'':''MEETINGINTERACTEDCOUNT'',''Assigned Products'':''ASSIGNEDPRODUCTS'',''Report Period'':''REPORTPERIOD''}, inplace =True)

result\_email = session.write\_pandas(graph\_data\_df,"API\_EMAIL\_DATA",database = "RAW\_DEV",schema="DEV\_RAW\_MICROSOFTGRAPH")

row\_count\_email = len(graph\_data\_df.index) # Adjust count for lists

# Insert audit log (Success)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_EMAIL\_DATA'',?, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''SUCCESS'', NULL)

""", [row\_count\_email]).collect()

else:

raise Exception(f"Error calling Microsoft Graph API: {response\_email.text}")

except Exception as e:

error\_message = str(e)

# Insert audit log (Failure)

session.sql("""

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT(SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_DEV'', ''DEV\_RAW\_MICROSOFTGRAPH'', ''API\_EMAIL\_DATA'',NULL, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''SUCCESS'', ?)

""", [error\_message]).collect()

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_DEV.DEV\_RAW\_MERAKI.EMAIL WHERE FLAG=TRUE;"

email\_result = session.sql(email\_query).collect()

email\_list = [row["EMAIL\_ID"] for row in email\_result]

if email\_list:

email\_addresses = ",".join(email\_list)

# Send email notification

session.sql("""

CALL SYSTEM$SEND\_EMAIL(

''email\_integration'',

?,

?,

?

)

""", [email\_addresses, f"{location} Microsoft Graph API Failed", error\_message]).collect()

return f"Procedure failed: {error\_message}"

return "Data inserted successfully"

';

**Load Flattened data from RAW to HARMONIZED Layer:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS

$$

DECLARE RecordCount\_Employee NUMBER;

RecordCount\_Email NUMBER;

RecordCount\_Intune NUMBER;

TableName VARCHAR;

BEGIN

-- Author: Bharath Sridhar

-- Created: 02/14/2025

-- Description: Continuous load Procedure

-- Modifications: 02/14/2025: Created the procedure for Microsoft Continuous Load from RAW to HARMONIZED database

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.EMPLOYEES(

OBJECTID,

FIRSTNAME,

LASTNAME,

DISPLAYNAME,

EMAIL,

DEPARTMENT,

OFFICELOCATION,

EMPLOYEETYPE,

HIREDATE,

ACCOUNTENABLED

)

SELECT

t0.value:"id"::varchar,

t0.value:"givenName"::varchar,

t0.value:"surname"::varchar,

t0.value:"displayName"::varchar,

t0.value:"userPrincipalName"::varchar,

t0.value:"department"::varchar,

t0.value:"officeLocation"::varchar,

t0.value:"employeeType"::varchar,

t0.value:"employeeHireDate"::DATE,

t0.value:"accountEnabled"::boolean

FROM

RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_EMPLOYEE\_DATA,

LATERAL FLATTEN( INPUT => DATA:value ) as t0;

TableName := 'EMPLOYEES';

RecordCount\_Employee := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'EMPLOYEES',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_CONTINUOUS\_LOAD',

'SUCCESS',

NULL

);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.EMAIL\_ACTIVITY(

EMAILACTIVITYID,

OBJECTID,

EMAIL,

REPORTREFRESHDATE,

DISPLAYNAME,

SENDCOUNT,

RECEIVECOUNT,

MEETINGCREATEDCOUNT,

MEETINGINTERACTEDCOUNT,

REPORTPERIOD,

DATASOURCE,

ISDELETED,

DELETEDDATE,

LASTACTIVITYDATE,

ASSIGNEDPRODUCTS,

READCOUNT

)

SELECT

NULL,

NULL,

USERPRINCIPALNAME,

REPORTREFRESHDATE,

DISPLAYNAME,

SENDCOUNT,

RECEIVECOUNT,

MEETINGCREATEDCOUNT,

MEETINGINTERACTEDCOUNT,

REPORTPERIOD,

NULL,

ISDELETED,

DELETEDDATE,

LASTACTIVITYDATE,

ASSIGNEDPRODUCTS,

READCOUNT

FROM

RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_EMAIL\_DATA;

TableName := 'EMAIL\_ACTIVITY';

RecordCount\_Email := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'EMAIL\_ACTIVITY',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_CONTINUOUS\_LOAD',

'SUCCESS',

NULL

);

INSERT INTO HARMONIZED\_DEV.DEV\_HAR\_MICROSOFTGRAPH.INTUNE\_DATA(

ATTENDANCEDATE,

ACTIVATIONLOCKBYPASSCODE,

ANDROIDSECURITYPATCHLEVEL,

AZUREADDEVICEID,

AZUREADREGISTERED,

COMPLIANCEGRACEPERIODEXPIRATIONDATETIME,

COMPLIANCESTATE,

CONFIGURATIONMANAGERCLIENTENABLEDFEATURES,

DEVICEACTIONRESULTS,

DEVICECATEGORYDISPLAYNAME,

DEVICEENROLLMENTTYPE,

DEVICEHEALTHATTESTATIONSTATE,

DEVICENAME,

DEVICEREGISTRATIONSTATE,

EASACTIVATED,

EASACTIVATIONDATETIME,

EASDEVICEID,

EMAILADDRESS,

ENROLLEDDATETIME,

ENROLLMENTPROFILENAME,

ETHERNETMACADDRESS,

EXCHANGEACCESSSTATE,

EXCHANGEACCESSSTATEREASON,

EXCHANGELASTSUCCESSFULSYNCDATETIME,

FREESTORAGESPACEINBYTES,

ICCID,

ID,

IMEI,

ISENCRYPTED,

ISSUPERVISED,

JAILBROKEN,

LASTSYNCDATETIME,

MANAGEDDEVICENAME,

MANAGEDDEVICEOWNERTYPE,

MANAGEMENTAGENT,

MANAGEMENTCERTIFICATEEXPIRATIONDATE,

MANUFACTURER,

MEID,

MODEL,

NOTES,

OPERATINGSYSTEM,

OSVERSION,

PARTNERREPORTEDTHREATSTATE,

PHONENUMBER,

PHYSICALMEMORYINBYTES,

REMOTEASSISTANCESESSIONERRORDETAILS,

REMOTEASSISTANCESESSIONURL,

REQUIREUSERENROLLMENTAPPROVAL,

SERIALNUMBER,

SUBSCRIBERCARRIER,

TOTALSTORAGESPACEINBYTES,

UDID,

USERDISPLAYNAME,

USERID,

USERPRINCIPALNAME,

WIFIMACADDRESS

)

SELECT

current\_timestamp::timestamp\_ntz,

t0.value:"activationLockBypassCode"::varchar,

t0.value:"androidSecurityPatchLevel"::varchar,

t0.value:"azureADDeviceId"::varchar,

t0.value:"azureADRegistered"::boolean,

t0.value:"complianceGracePeriodExpirationDateTime"::timestamp,

t0.value:"complianceState"::varchar,

t0.value:"configurationManagerClientEnabledFeatures"::varchar,

t0.value:"deviceActionResults"::varchar,

t0.value:"deviceCategoryDisplayName"::varchar,

t0.value:"deviceEnrollmentType"::varchar,

t0.value:"deviceHealthAttestationState"::varchar,

t0.value:"deviceName"::varchar,

t0.value:"deviceRegistrationState"::varchar,

t0.value:"easActivated"::boolean,

t0.value:"easActivationDateTime"::timestamp,

t0.value:"easDeviceId"::varchar,

t0.value:"emailAddress"::varchar,

t0.value:"enrolledDateTime"::timestamp,

t0.value:"enrollmentProfileName"::varchar,

t0.value:"ethernetMacAddress"::varchar,

t0.value:"exchangeAccessState"::varchar,

t0.value:"exchangeAccessStateReason"::varchar,

t0.value:"exchangeLastSuccessfulSyncDateTime"::timestamp,

t0.value:"freeStorageSpaceInBytes"::number,

t0.value:"iccid"::varchar,

t0.value:"id"::varchar,

t0.value:"imei"::varchar,

t0.value:"isEncrypted"::boolean,

t0.value:"isSupervised"::boolean,

t0.value:"jailBroken"::varchar,

t0.value:"lastSyncDateTime"::timestamp,

t0.value:"managedDeviceName"::varchar,

t0.value:"managedDeviceOwnerType"::varchar,

t0.value:"managementAgent"::varchar,

t0.value:"managementCertificateExpirationDate"::timestamp,

t0.value:"manufacturer"::varchar,

t0.value:"meid"::varchar,

t0.value:"model"::varchar,

t0.value:"notes"::varchar,

t0.value:"operatingSystem"::varchar,

t0.value:"osVersion"::varchar,

t0.value:"partnerReportedThreatState"::varchar,

t0.value:"phoneNumber"::varchar,

t0.value:"physicalMemoryInBytes"::number,

t0.value:"remoteAssistanceSessionErrorDetails"::varchar,

t0.value:"remoteAssistanceSessionUrl"::varchar,

t0.value:"requireUserEnrollmentApproval"::boolean,

t0.value:"serialNumber"::varchar,

t0.value:"subscriberCarrier"::varchar,

t0.value:"totalStorageSpaceInBytes"::number,

t0.value:"udid"::varchar,

t0.value:"userDisplayName"::varchar,

t0.value:"userId"::varchar,

t0.value:"userPrincipalName"::varchar,

t0.value:"wiFiMacAddress"::varchar

FROM

RAW\_DEV.DEV\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA,

LATERAL FLATTEN( INPUT => DATA:value ) as t0;

TableName := 'INTUNE\_DATA';

RecordCount\_Intune := SQLROWCOUNT;

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

'INTUNE\_DATA',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_CONTINUOUS\_LOAD',

'SUCCESS',

NULL

);

EXCEPTION

WHEN OTHER THEN

INSERT INTO HARMONIZED\_DEV.AUDIT\_LOGS.AUDIT (

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

'MICROSOFT GRAPH',

'HARMONIZED\_DEV',

'DEV\_HAR\_MICROSOFTGRAPH',

:TableName,

0,

CURRENT\_TIMESTAMP(),

'MICROSOFT\_CONTINUOUS\_LOAD',

'FAILURE',

:sqlerrm

);

END;

$$;