**ATTENDANCE DATA PROD STORED PROCEDURES**

**Meraki RAW Historical Load:**

CREATE OR REPLACE PROCEDURE RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_HISTORICAL\_LOAD\_PROC()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE RecordCount INT;

BEGIN

-- Author: VAMSI

-- Created: 17/03/2025

-- Description: Historical load Procedure

-- Modifications: 17/02/2025: Created the procedure for M Historical Load

COPY INTO RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_HISTORICAL FROM ''@"RAW\_PROD"."PROD\_RAW\_MERAKI"."MERAKI\_STAGE"/meraki\_data.csv'' FILE\_FORMAT = (TYPE = ''CSV'' FIELD\_OPTIONALLY\_ENCLOSED\_BY = ''"'' SKIP\_HEADER = 1) ON\_ERROR = ''SKIP\_FILE'';

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_API'',

''RAW\_PROD'',

''PROD\_RAW\_MERAKI'',

''MERAKI\_HISTORICAL'',

:RecordCount,

CURRENT\_TIMESTAMP(),

''MERAKI\_HISTORICAL\_LOAD\_PROC'',

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_API'',

''RAW\_PROD'',

''PROD\_RAW\_MERAKI'',

''MERAKI\_HISTORICAL'',

0,

CURRENT\_TIMESTAMP(),

''MERAKI\_HISTORICAL\_LOAD\_PROC'',

''ERROR'',

:sqlerrm

);

END;

';



**Meraki RAW Continuous Load:**

CREATE OR REPLACE PROCEDURE RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_API\_CONTINUOUSLOAD()

RETURNS VARCHAR

LANGUAGE PYTHON

RUNTIME\_VERSION = '3.10'

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

HANDLER = 'fetch\_posts'

EXTERNAL\_ACCESS\_INTEGRATIONS = (MERAKI\_EXTERNAL\_NETWORK\_ACCESS)

SECRETS = ('cred'=MERAKI\_SECRET)

EXECUTE AS OWNER

AS '

import \_snowflake

import requests

import json

from snowflake.snowpark import Session

def fetch\_posts(session: Session):

token = \_snowflake.get\_generic\_secret\_string(''cred'')

api\_urls = {

"San\_Francisco": "https://api.meraki.com/api/v1/networks/L\_638385247179776905/clients?perPage=1000&vlan=108",

"San\_Juan": "https://api.meraki.com/api/v1/networks/L\_638385247179774525/clients?perPage=1000&vlan=88",

"New\_York\_City": "https://api.meraki.com/api/v1/networks/L\_638385247179774894/clients?perPage=1000&vlan=78"

}

headers = {"Authorization": f"Bearer {token}"}

# Ensure raw table exists

session.sql("CREATE TABLE IF NOT EXISTS RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_RAW (data VARIANT)").collect()

# Ensure audit table exists

session.sql("""

CREATE TABLE IF NOT EXISTS HARMONIZED\_PROD.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()

success\_flag = True

error\_messages = []

for location, url in api\_urls.items():

try:

response = requests.get(url, headers=headers)

if response.status\_code == 200:

json\_data = response.json()

row\_count = len(json\_data) if isinstance(json\_data, list) else 1

# Insert data into raw table

session.sql(

"INSERT INTO RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_RAW (data) SELECT PARSE\_JSON(?)",

[json.dumps(json\_data)]

).collect()

# Insert audit log (Success)

session.sql("""

INSERT INTO HARMONIZED\_PROD.AUDIT\_LOGS.AUDIT

(SOURCE, DATABASE\_NAME, SCHEMA\_NAME, TABLE\_NAME, ROW\_COUNT, LOAD\_TIMESTAMP, PROCEDURE\_NAME, STATUS, ERROR\_MESSAGE)

VALUES (?, ''RAW\_PROD'', ''PROD\_RAW\_MERAKI'', ''MERAKI\_RAW'', ?, CURRENT\_TIMESTAMP(), ''Meraki\_API\_ContinuousLoad'', ''SUCCESS'', NULL)

""", [location + "\_MERAKI\_API", row\_count]).collect()

else:

raise Exception(f"Error fetching {location} data: {response.status\_code}. Response: {response.text}")

except Exception as e:

success\_flag = False

error\_message = str(e)

error\_messages.append(f"{location}: {error\_message}")

# Insert audit log (Failure)

session.sql("""

INSERT INTO HARMONIZED\_PROD.AUDIT\_LOGS.AUDIT

(SOURCE, DATABASE\_NAME, SCHEMA\_NAME, TABLE\_NAME, ROW\_COUNT, LOAD\_TIMESTAMP, PROCEDURE\_NAME, STATUS, ERROR\_MESSAGE)

VALUES (?, ''RAW\_PROD'', ''PROD\_RAW\_MERAKI'', ''MERAKI\_RAW'', NULL, CURRENT\_TIMESTAMP(), ''Meraki\_API\_ContinuousLoad'', ''FAILURE'', ?)

""", [location + "\_MERAKI\_API", error\_message]).collect()

# Get email recipients

email\_query = "SELECT EMAIL\_ID FROM RAW\_PROD.PROD\_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} Meraki API Failed", error\_message]).collect()

# Return final status based on all API calls

if success\_flag:

return "Procedure executed successfully"

else:

return f"Procedure failed. Errors: {'' | ''.join(error\_messages)}"

';



**MS Graph RAW Historical Load:**

CREATE OR REPLACE PROCEDURE RAW\_PROD.PROD\_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\_PROD.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)

);

TableName := ''ACCESS\_CARDS'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''ACCESS\_CARDS'',

:RecordCount\_Access,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

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

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

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

:RecordCount\_AccessEmail,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''DEVICES'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''DEVICES'',

:RecordCount\_Devices,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''EMAIL\_ACTIVITY'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''EMAIL\_ACTIVITY'',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''EMPLOYEES'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''EMPLOYEES'',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''INTUNE\_DATA'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''INTUNE\_DATA'',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''LOCATIONS'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''LOCATIONS'',

:RecordCount\_Location,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

TableName := ''VISITS'';

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

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

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

''VISITS'',

:RecordCount\_Visits,

CURRENT\_TIMESTAMP(),

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

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFT GRAPH'',

''RAW\_PROD'',

''PROD\_RAW\_MICROSOFTGRAPH'',

:TableName,

0,

CURRENT\_TIMESTAMP(),

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

''FAILURE'',

:SQLERRM

);

END;

';



**MS Graph RAW Continuous Load:**

CREATE OR REPLACE PROCEDURE RAW\_PROD.PROD\_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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA (data VARIANT)").collect()

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

# Ensure audit table exists

session.sql("""

CREATE TABLE IF NOT EXISTS HARMONIZED\_PROD.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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA (data) SELECT PARSE\_JSON(?)", [json.dumps(graph\_data\_intune)]).collect()

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

# Insert audit log (Success)

session.sql("""

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

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_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\_PROD.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_RAW\_MICROSOFTGRAPH'', ''API\_INTUNE\_DATA'',NULL, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''FAILED'', ?)

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

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_PROD.PROD\_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" Intune Microsoft Graph API Failed", error\_message]).collect()

################ 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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_EMPLOYEE\_DATA (data) SELECT PARSE\_JSON(?)", [json.dumps(graph\_data\_employee)]).collect()

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

# Insert audit log (Success)

session.sql("""

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

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_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\_PROD.AUDIT\_LOGS.AUDIT (SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_RAW\_MICROSOFTGRAPH'', ''API\_EMPLOYEE\_DATA'', NULL , CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''FAILED'', ?)

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

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_PROD.PROD\_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"Employee Microsoft Graph API Failed", error\_message]).collect()

################ 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\_PROD",schema="PROD\_RAW\_MICROSOFTGRAPH")

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

# Insert audit log (Success)

session.sql("""

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

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_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\_PROD.AUDIT\_LOGS.AUDIT(SOURCE,DATABASE\_NAME,SCHEMA\_NAME,TABLE\_NAME,ROW\_COUNT,LOAD\_TIMESTAMP,PROCEDURE\_NAME,STATUS,ERROR\_MESSAGE)

VALUES (''Mirosoft Graph API'',''RAW\_PROD'', ''PROD\_RAW\_MICROSOFTGRAPH'', ''API\_EMAIL\_DATA'',NULL, CURRENT\_TIMESTAMP(), ''Microsoft\_GraphAPI\_ContinuousLoad'', ''FAILED'', ?)

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

# Send email notification

email\_query = "SELECT EMAIL\_ID FROM RAW\_PROD.PROD\_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"Email Microsoft Graph API Failed", error\_message]).collect()

return "Data inserted successfully"

';



**Meraki HARMONIZED Historical Load:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_PROD.PROD\_HAR\_MERAKI.MERAKI\_HISTORICAL\_HAR()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE RecordCount\_Meraki NUMBER;

TableName STRING;

BEGIN

-- Author: Vamsi

-- Created: 02/17/2025

-- Description: Historical load Procedure

-- Modifications: 02/17/2025: Created the procedure for Meraki Historical Load to Harmonized DB

CREATE TABLE IF NOT EXISTS HARMONIZED\_PROD.PROD\_HAR\_MERAKI.MERAKI

(

adaptivePolicyGroup VARCHAR(50),

description VARCHAR(100),

deviceTypePrediction VARCHAR(50),

firstSeen TIMESTAMP\_NTZ(9),

groupPolicy8021x VARCHAR(50),

id VARCHAR(30),

ip VARCHAR(40),

ip6 VARCHAR(50),

ip6Local VARCHAR(50),

lastSeen TIMESTAMP\_NTZ(9),

mac VARCHAR(20),

manufacturer VARCHAR(30),

namedVlan VARCHAR(20),

notes VARCHAR(100),

os VARCHAR(30),

pskGroup VARCHAR(30),

recentDeviceConnection VARCHAR(50),

recentDeviceMac VARCHAR(30),

recentDeviceName VARCHAR(20),

recentDeviceSerial VARCHAR(30),

smInstalled VARCHAR(20),

ssid VARCHAR(30),

status VARCHAR(20),

switchport VARCHAR(20),

usage\_RECV NUMBER,

usage\_SENT NUMBER,

usage\_TOTAL NUMBER,

user VARCHAR(30),

vlan NUMBER,

wirelessCapabilities VARCHAR(50),

LOCATION VARCHAR(20),

ATTENDANCEDATE TIMESTAMP\_NTZ(9)

);

CREATE TABLE IF NOT EXISTS HARMONIZED\_PROD.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\_PROD.PROD\_HAR\_MERAKI.MERAKI (

adaptivePolicyGroup,

description,

deviceTypePrediction,

firstSeen,

groupPolicy8021x,

id,

ip,

ip6,

ip6Local,

lastSeen,

mac,

manufacturer,

namedVlan,

notes,

os,

pskGroup,

recentDeviceConnection,

recentDeviceMac,

recentDeviceName,

recentDeviceSerial,

smInstalled,

ssid,

status,

switchport,

usage\_RECV ,

usage\_SENT ,

usage\_TOTAL,

user,

vlan,

wirelessCapabilities,

LOCATION,

ATTENDANCEDATE

)

SELECT

NULL,

DESCRIPTION,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

LASTSEEN,

MACADDRESS,

NULL,

NULL,

NULL,

NULL,

NULL,

RECENTDEVICECONNECTION,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

NULL,

LOCATION,

ATTENDANCEDATE

FROM RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_HISTORICAL;

TableName := ''MERAKI'';

RecordCount\_Meraki := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MERAKI'',

''MERAKI'',

:RecordCount\_Meraki,

CURRENT\_TIMESTAMP(),

''MERAKI\_HISTORICAL\_HAR'',

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MERAKI'',

''MERAKI'',

0,

CURRENT\_TIMESTAMP(),

''MERAKI\_HISTORICAL\_HAR'',

''FAILURE'',

:sqlerrm);

END;

';



**Meraki HARMONIZED Flattening:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_PROD.PROD\_HAR\_MERAKI.MERAKI\_TRANSFORMATION()

RETURNS VARCHAR(500)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE RecordCount\_Meraki NUMBER;

TableName VARCHAR;

email\_list STRING;

email\_count NUMBER;

BEGIN

-- Author: Vamsi

-- 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\_PROD.PROD\_HAR\_MERAKI.MERAKI(

adaptivePolicyGroup,

description,

deviceTypePrediction,

firstSeen,

groupPolicy8021x,

id,

ip,

ip6,

ip6Local,

lastSeen,

mac,

manufacturer,

namedVlan,

notes,

os,

pskGroup,

recentDeviceConnection,

recentDeviceMac,

recentDeviceName,

recentDeviceSerial,

smInstalled,

ssid,

status,

switchport,

usage\_RECV ,

usage\_SENT ,

usage\_TOTAL,

user,

vlan,

wirelessCapabilities,

location,

attendancedate

)

SELECT

flattened\_data.value:adaptivePolicyGroup::STRING AS adaptivePolicyGroup,

flattened\_data.value:description::STRING AS description,

flattened\_data.value:deviceTypePrediction::STRING AS deviceTypePrediction,

flattened\_data.value:firstSeen::STRING AS firstSeen,

flattened\_data.value:groupPolicy8021x::STRING AS groupPolicy8021x,

flattened\_data.value:id::STRING AS id,

flattened\_data.value:ip::STRING AS ip,

flattened\_data.value:ip6::STRING AS ip6,

flattened\_data.value:ip6Local::STRING AS ip6Local,

flattened\_data.value:lastSeen::STRING AS lastSeen,

replace(flattened\_data.value:mac::STRING, '':'','''') AS mac,

flattened\_data.value:manufacturer::STRING AS manufacturer,

flattened\_data.value:namedVlan::STRING AS namedVlan,

flattened\_data.value:notes::STRING AS notes,

flattened\_data.value:os::STRING AS os,

flattened\_data.value:pskGroup::STRING AS pskGroup,

flattened\_data.value:recentDeviceConnection::STRING AS recentDeviceConnection,

flattened\_data.value:recentDeviceMac::STRING AS recentDeviceMac,

flattened\_data.value:recentDeviceName::STRING AS recentDeviceName,

flattened\_data.value:recentDeviceSerial::STRING AS recentDeviceSerial,

flattened\_data.value:smInstalled::STRING AS smInstalled,

flattened\_data.value:ssid::STRING AS ssid,

flattened\_data.value:status::STRING AS status,

flattened\_data.value:switchport::STRING AS switchport,

flattened\_data.value:usage.recv::NUMBER AS recv,

flattened\_data.value:usage.sent::NUMBER AS sent,

flattened\_data.value:usage.total::NUMBER AS total,

flattened\_data.value:user::STRING AS user,

flattened\_data.value:vlan::STRING AS vlan,

flattened\_data.value:wirelessCapabilities::STRING AS wirelessCapabilities,

case

when flattened\_data.value:vlan::NUMBER = 108 THEN ''San Francisco''

when flattened\_data.value:vlan::NUMBER = 88 THEN ''San Juan''

when flattened\_data.value:vlan::NUMBER = 78 THEN ''New York City''

ELSE NULL

END AS location,

CURRENT\_TIMESTAMP as attendancedate

FROM

RAW\_PROD.PROD\_RAW\_MERAKI.MERAKI\_RAW,

LATERAL FLATTEN(input => MERAKI\_RAW.DATA) AS flattened\_data;

TableName := ''MERAKI'';

RecordCount\_Meraki := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MERAKI'',

''MERAKI'',

:RecordCount\_Meraki,

CURRENT\_TIMESTAMP(),

''MERAKI\_TRANSFORMATION'',

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

SELECT COUNT(\*) INTO :email\_count FROM RAW\_PROD.PROD\_RAW\_MERAKI.EMAIL WHERE flag = TRUE;

IF (:email\_count > 0) THEN

SELECT ARRAY\_TO\_STRING(ARRAY\_AGG(email\_id),'','')

into email\_list

from RAW\_PROD.PROD\_RAW\_MERAKI.EMAIL

where flag =TRUE;

call system$send\_email(

''email\_integration'',

:email\_list,

''Email Alert: Error while executing MERAKI\_TRANSFORMATION Procedure'',

:sqlerrm

);

END IF;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MERAKI\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MERAKI'',

''MERAKI'',

0,

CURRENT\_TIMESTAMP(),

''MERAKI\_TRANSFORMATION'',

''FAILURE'',

:sqlerrm

);

END;

';



**MS Graph HARMONIZED Historical Load:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.MICROSOFT\_HISTORICAL\_LOAD\_HAR()

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\_PROD.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)

);

TableName := ''ACCESS\_CARDS'';

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.ACCESS\_CARDS (

ID,

USERNAME,

VISIT\_DATE,

LOCATION,

EMAIL

)

SELECT

ID,

USERNAME,

VISIT\_DATE,

LOCATION,

EMAIL

FROM RAW\_PROD.PROD\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS;

RecordCount\_Access := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''ACCESS\_CARDS'',

:RecordCount\_Access,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL

);

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

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.ACCESS\_CARDS\_EMAIL\_MAP(

INDEX,

USERNAME,

EMAIL

)

SELECT

INDEX,

USERNAME,

EMAIL

FROM RAW\_PROD.PROD\_RAW\_MICROSOFTGRAPH.ACCESS\_CARDS\_EMAIL\_MAP;

RecordCount\_AccessEmail := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

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

:RecordCount\_AccessEmail,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL

);

TableName := ''DEVICES'';

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.DEVICES(

DEVICEID,

OPERATINGSYSTEM,

MACADDRESS,

EMPLOYEEEMAIL

)

SELECT

DEVICEID,

OPERATINGSYSTEM,

MACADDRESS,

EMPLOYEEEMAIL

FROM RAW\_PROD.PROD\_RAW\_MICROSOFTGRAPH.DEVICES;

RecordCount\_Devices := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''DEVICES'',

:RecordCount\_Devices,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

TableName := ''EMAIL\_ACTIVITY'';

INSERT INTO HARMONIZED\_PROD.PROD\_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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.EMAIL\_ACTIVITY;

RecordCount\_Email := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''EMAIL\_ACTIVITY'',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

TableName := ''EMPLOYEES'';

INSERT INTO HARMONIZED\_PROD.PROD\_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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.EMPLOYEES;

RecordCount\_Employee := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''EMPLOYEES'',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

TableName := ''INTUNE\_DATA'';

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.INTUNE\_DATA(

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

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,

ATTENDANCEDATE,

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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.INTUNE\_DATA;

RecordCount\_Intune := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''INTUNE\_DATA'',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

TableName := ''LOCATIONS'';

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.LOCATIONS(

LOCATIONID,

LOCATIONNAME,

VLAN

)

SELECT

LOCATIONID,

LOCATIONNAME,

VLAN

FROM RAW\_PROD.PROD\_RAW\_MICROSOFTGRAPH.LOCATIONS;

RecordCount\_Location := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''LOCATIONS'',

:RecordCount\_Location,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

TableName := ''VISITS'';

INSERT INTO HARMONIZED\_PROD.PROD\_HAR\_MICROSOFTGRAPH.VISITS(

VISITID,

EMPLOYEEEMAIL,

LOCATIONNAME,

VISITDATETIME,

VISITDATE,

MACADDRESS

)

SELECT

VISITID,

EMPLOYEEEMAIL,

LOCATIONNAME,

VISITDATETIME,

VISITDATE,

MACADDRESS

FROM RAW\_PROD.PROD\_RAW\_MICROSOFTGRAPH.VISITS;

RecordCount\_Visits := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''VISITS'',

:RecordCount\_Visits,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''SUCCESS'',

NULL);

EXCEPTION

WHEN OTHER THEN

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

:TableName,

0,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_HISTORICAL\_LOAD\_HAR'',

''FAILURE'',

:sqlerrm);

END;

';



**MS Graph HARMONIZED Flattening:**

CREATE OR REPLACE PROCEDURE HARMONIZED\_PROD.PROD\_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;

email\_list STRING;

email\_count NUMBER;

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

TableName := ''EMPLOYEES'';

INSERT INTO HARMONIZED\_PROD.PROD\_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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_EMPLOYEE\_DATA,

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

RecordCount\_Employee := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''EMPLOYEES'',

:RecordCount\_Employee,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN'',

''SUCCESS'',

NULL

);

TableName := ''EMAIL\_ACTIVITY'';

INSERT INTO HARMONIZED\_PROD.PROD\_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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_EMAIL\_DATA;

RecordCount\_Email := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''EMAIL\_ACTIVITY'',

:RecordCount\_Email,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN'',

''SUCCESS'',

NULL

);

TableName := ''INTUNE\_DATA'';

INSERT INTO HARMONIZED\_PROD.PROD\_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,

REPLACE(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\_PROD.PROD\_RAW\_MICROSOFTGRAPH.API\_INTUNE\_DATA,

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

RecordCount\_Intune := SQLROWCOUNT;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

''INTUNE\_DATA'',

:RecordCount\_Intune,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN'',

''SUCCESS'',

NULL

);

EXCEPTION

WHEN OTHER THEN

SELECT COUNT(\*) INTO :email\_count FROM RAW\_PROD.PROD\_RAW\_MERAKI.EMAIL WHERE flag = TRUE;

IF (:email\_count > 0) THEN

SELECT ARRAY\_TO\_STRING(ARRAY\_AGG(email\_id),'','')

into email\_list

from RAW\_PROD.PROD\_RAW\_MERAKI.EMAIL

where flag =TRUE;

call system$send\_email(

''email\_integration'',

:email\_list,

''Email Alert: Error while executing MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN Procedure'',

:sqlerrm

);

END IF;

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

SOURCE,

DATABASE\_NAME,

SCHEMA\_NAME,

TABLE\_NAME,

ROW\_COUNT,

LOAD\_TIMESTAMP,

PROCEDURE\_NAME,

STATUS,

ERROR\_MESSAGE

)

VALUES(

''MICROSOFTGRAPH\_RAW\_DB'',

''HARMONIZED\_PROD'',

''PROD\_HAR\_MICROSOFTGRAPH'',

:TableName,

0,

CURRENT\_TIMESTAMP(),

''MICROSOFT\_CONTINUOUS\_LOAD\_FLATTEN'',

''FAILURE'',

:sqlerrm

);

END;

';

