Python Script for Employee Master API

!pip install azure-keyvault

import pandas as pd

import pyodbc

import numpy as np

import json

from azure.identity import ClientSecretCredential

from azure.keyvault.secrets import SecretClient

from azure.identity import ClientSecretCredential

import time

from datetime import datetime, timedelta

KEYVAULT\_URI = "https://std-inc-hr-kv-01.vault.azure.net/"

tenant\_id = '1250f2eb-4784-4223-98dc-d6e334455123vc'

client\_id = '969c6875-c128-4a09-9d94-66c8f216b4rf5'

client\_secret = '~PX8Q~YzQ\_WjJr1cmd8SawqdpChBAjwUbDmnC312dvO'

credential = ClientSecretCredential(tenant\_id=tenant\_id, client\_id=client\_id, client\_secret=client\_secret)

client = SecretClient(vault\_url=KEYVAULT\_URI, credential=credential)

DEMO\_DB\_PASSWORD=client.get\_secret("STD-INC-HR-SQLDB-01-SK-01").value

server = 'std-inc-hr-sqldbsrv-01.database.windows.net'

db1 = 'STD-INC-HR-SQLDB-01'

username = 'hrsqladmin'

password = DEMO\_DB\_PASSWORD

cnxn = pyodbc.connect('DRIVER={ODBC Driver 18 for SQL Server};SERVER='+server+';DATABASE='+db1+';ENCRYPT=no;UID='+username+';PWD='+ password)

cur = cnxn.cursor()

today = datetime.today()

yesterday = today - timedelta(days=1)

yesterday=yesterday.strftime('%d-%m-%Y')

yesterday

#for active employee

import requests

url = "https://myjsw.darwinbox.in/masterapi/employee"

payload = '{\n    "api\_key":"e121263966411928e7ba52c0c92823af30b12afa928fcac9b6925ab8bc79cb49a71b5846f91a2119b07e07220871057d5ddd2756f18470bde3ea691acxv45",\n    "datasetKey":"8056f266cbea22b92af8ab599761f8912ebaa8e816eb5159e71d448ddf1b43f43759a69d8c50ef4abc0e326feb2c5e15ec67f4c2ebdaad840421wsd5vef4e3c",\n"last\_modified":"%s 12:00:00"\n\n}'%(yesterday)

headers = {

  'Authorization': 'Basic QVBJX1VzZXJfdm9rc2Vfb3V0OnI5cTdPOXlfTno4WTE4JldIjksqwrxT',

  'Content-Type': 'text/plain',

}

response1 = requests.request("POST", url, headers=headers, data=payload)

#print(response1.text)

df1 = pd.DataFrame(response1.json())

a1 = []

for x in df1["employee\_data"]:

    a1.append(x)

df1=pd.DataFrame(a1)

#df1

df1.replace("N.A",np.NaN, inplace = True)

df1["division"] = df1["division"].fillna("-")

df1["date\_of\_joining"]=pd.to\_datetime(df1["date\_of\_joining"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["date\_of\_exit"]=pd.to\_datetime(df1["date\_of\_exit"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["date\_of\_birth"]=pd.to\_datetime(df1["date\_of\_birth"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["date\_of\_resignation"]=pd.to\_datetime(df1["date\_of\_resignation"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["date\_of\_confirmation"]=pd.to\_datetime(df1["date\_of\_confirmation"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["activation\_timestamp"]=pd.to\_datetime(df1["activation\_timestamp"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["group\_date\_of\_joining"]=pd.to\_datetime(df1["group\_date\_of\_joining"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["cost\_center\_effective\_from"]=pd.to\_datetime(df1["cost\_center\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["designation\_title\_effective\_from"]=pd.to\_datetime(df1["designation\_title\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["direct\_manager\_effective\_from"]=pd.to\_datetime(df1["direct\_manager\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["employee\_type\_effective\_from"]=pd.to\_datetime(df1["employee\_type\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["functional\_area\_effective\_from"]=pd.to\_datetime(df1["functional\_area\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["job\_level\_effective\_from"]=pd.to\_datetime(df1["job\_level\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["latest\_office\_effective\_from"]=pd.to\_datetime(df1["latest\_office\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["role\_effective\_from"]=pd.to\_datetime(df1["role\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df1["date\_of\_joining"]=pd.to\_datetime(df1["date\_of\_joining"],format='%d-%m-%Y')

df1["date\_of\_exit"]=pd.to\_datetime(df1["date\_of\_exit"],format='%d-%m-%Y')

df1["date\_of\_birth"]=pd.to\_datetime(df1["date\_of\_birth"],format='%d-%m-%Y')

df1["date\_of\_resignation"]=pd.to\_datetime(df1["date\_of\_resignation"],format='%d-%m-%Y')

df1["date\_of\_confirmation"]=pd.to\_datetime(df1["date\_of\_confirmation"],format='%d-%m-%Y')

df1["activation\_timestamp"]=pd.to\_datetime(df1["activation\_timestamp"],format='%d-%m-%Y')

df1["group\_date\_of\_joining"]=pd.to\_datetime(df1["group\_date\_of\_joining"],format='%d-%m-%Y')

df1["cost\_center\_effective\_from"]=pd.to\_datetime(df1["cost\_center\_effective\_from"],format='%d-%m-%Y')

df1["designation\_title\_effective\_from"]=pd.to\_datetime(df1["designation\_title\_effective\_from"],format='%d-%m-%Y')

df1["direct\_manager\_effective\_from"]=pd.to\_datetime(df1["direct\_manager\_effective\_from"],format='%d-%m-%Y')

df1["employee\_type\_effective\_from"]=pd.to\_datetime(df1["employee\_type\_effective\_from"],format='%d-%m-%Y')

df1["functional\_area\_effective\_from"]=pd.to\_datetime(df1["functional\_area\_effective\_from"],format='%d-%m-%Y')

df1["job\_level\_effective\_from"]=pd.to\_datetime(df1["job\_level\_effective\_from"],format='%d-%m-%Y')

df1["latest\_office\_effective\_from"]=pd.to\_datetime(df1["latest\_office\_effective\_from"],format='%d-%m-%Y')

df1["role\_effective\_from"]=pd.to\_datetime(df1["role\_effective\_from"],format='%d-%m-%Y')

#for df1

col1 = ", ".join([str(i.replace('/','\_').replace('.','')) for i in df1.columns.tolist()])

#col1

cur.execute("Truncate TABLE Stagging\_Employee\_Master;")

cnxn.commit()

#insertion for df1(Active)

for i,row in df1.iterrows():

    sql = "INSERT INTO Stagging\_Employee\_Master (" +col1 + ") VALUES (" + "?,"\*(len(row)-1) + "?)"

    cur.execute(sql, tuple(row))

cnxn.commit()

cur.execute("Update\_Production\_Employee\_Master;")

cnxn.commit()

cur.execute("Update\_Production\_Employee\_Master\_Fact\_Table;")

cnxn.commit()

cur.execute("insert\_Production\_Functional\_Area\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Employee\_Type\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Job\_Level\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Office\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Band\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

#for Inactive employee

import requests

url = "https://myjsw.darwinbox.in/masterapi/employee"

payload = '{\n    "api\_key":"9deb875a046b6fc7db35dca88b7a115a3636e08c2d496be09127415c0816ad5d947a99476dde09b6e0337e08b0e84682a3f520c113901be8dd4128c90012345fs",\n    "datasetKey":"ac766511fbcf41e454cd11f31c5c3f85efc0ad48f02b02a1c81035618d688b6f1c2cee97d135d7f004e48b1867d0deae1d99320c751f08dfe6aa540cnjbadbf12",\n"last\_modified":"%s 12:00:00"\n\n}'%(yesterday)

headers = {

  'Authorization': 'Basic QVBJX1VzZXJfdm9rc2Vfb3V0OnI5cTdPOXlfTno4WTE4Kgv34a',

  'Content-Type': 'text/plain',

}

response2 = requests.request("POST", url, headers=headers, data=payload)

#print(response2.text)

df2 = pd.DataFrame(response2.json())

a2 = []

for x in df2["employee\_data"]:

    a2.append(x)

df2=pd.DataFrame(a2)

#df2

df2.replace("N.A",np.NaN, inplace = True)

df2["division"] = df2["division"].fillna("-")

df2["date\_of\_joining"]=pd.to\_datetime(df2["date\_of\_joining"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["date\_of\_exit"]=pd.to\_datetime(df2["date\_of\_exit"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["date\_of\_birth"]=pd.to\_datetime(df2["date\_of\_birth"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["date\_of\_resignation"]=pd.to\_datetime(df2["date\_of\_resignation"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["date\_of\_confirmation"]=pd.to\_datetime(df2["date\_of\_confirmation"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["activation\_timestamp"]=pd.to\_datetime(df2["activation\_timestamp"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["group\_date\_of\_joining"]=pd.to\_datetime(df2["group\_date\_of\_joining"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["cost\_center\_effective\_from"]=pd.to\_datetime(df2["cost\_center\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["designation\_title\_effective\_from"]=pd.to\_datetime(df2["designation\_title\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["direct\_manager\_effective\_from"]=pd.to\_datetime(df2["direct\_manager\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["employee\_type\_effective\_from"]=pd.to\_datetime(df2["employee\_type\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["functional\_area\_effective\_from"]=pd.to\_datetime(df2["functional\_area\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["job\_level\_effective\_from"]=pd.to\_datetime(df2["job\_level\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

#df2["latest\_office\_effective\_from"]=pd.to\_datetime(df2["latest\_office\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["role\_effective\_from"]=pd.to\_datetime(df2["role\_effective\_from"],dayfirst=True).dt.strftime('%d-%m-%Y')

df2["date\_of\_joining"]=pd.to\_datetime(df2["date\_of\_joining"],format='%d-%m-%Y')

df2["date\_of\_exit"]=pd.to\_datetime(df2["date\_of\_exit"],format='%d-%m-%Y')

df2["date\_of\_birth"]=pd.to\_datetime(df2["date\_of\_birth"],format='%d-%m-%Y')

df2["date\_of\_resignation"]=pd.to\_datetime(df2["date\_of\_resignation"],format='%d-%m-%Y')

df2["date\_of\_confirmation"]=pd.to\_datetime(df2["date\_of\_confirmation"],format='%d-%m-%Y')

df2["activation\_timestamp"]=pd.to\_datetime(df2["activation\_timestamp"],format='%d-%m-%Y')

df2["group\_date\_of\_joining"]=pd.to\_datetime(df2["group\_date\_of\_joining"],format='%d-%m-%Y')

df2["cost\_center\_effective\_from"]=pd.to\_datetime(df2["cost\_center\_effective\_from"],format='%d-%m-%Y')

df2["designation\_title\_effective\_from"]=pd.to\_datetime(df2["designation\_title\_effective\_from"],format='%d-%m-%Y')

df2["direct\_manager\_effective\_from"]=pd.to\_datetime(df2["direct\_manager\_effective\_from"],format='%d-%m-%Y')

df2["employee\_type\_effective\_from"]=pd.to\_datetime(df2["employee\_type\_effective\_from"],format='%d-%m-%Y')

df2["functional\_area\_effective\_from"]=pd.to\_datetime(df2["functional\_area\_effective\_from"],format='%d-%m-%Y')

df2["job\_level\_effective\_from"]=pd.to\_datetime(df2["job\_level\_effective\_from"],format='%d-%m-%Y')

#df2["latest\_office\_effective\_from"]=pd.to\_datetime(df2["latest\_office\_effective\_from"],format='%d-%m-%Y')

df2["role\_effective\_from"]=pd.to\_datetime(df2["role\_effective\_from"],format='%d-%m-%Y')

#for df2

col2 = ", ".join([str(i.replace('/','\_').replace('.','')) for i in df2.columns.tolist()])

#col2

cur.execute("Truncate TABLE Stagging\_Employee\_Master;")

cnxn.commit()

#insertion for df2(Inactive)

for i,row in df2.iterrows():

    sql = "INSERT INTO Stagging\_Employee\_Master (" +col2 + ") VALUES (" + "?,"\*(len(row)-1) + "?)"

    cur.execute(sql, tuple(row))

cnxn.commit()

cur.execute("Update\_Production\_Employee\_Master;")

cnxn.commit()

cur.execute("Update\_Production\_Employee\_Master\_Fact\_Table;")

cnxn.commit()

cur.execute("insert\_Production\_Functional\_Area\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Employee\_Type\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Job\_Level\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Office\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Role\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

cur.execute("insert\_Production\_Band\_Effective\_Employee\_Master\_Dimension;")

cnxn.commit()

### Update Production\_Job\_Level\_Effective\_Employee\_Master\_Dimension to insert Mapped Grades in Band Column

cur.execute("SP\_Update\_Band\_Column\_To\_Job\_Level;")

cnxn.commit()

cur.execute("SP\_Insert\_In\_Job\_level\_Update\_Band\_Sorting;")

cnxn.commit()