**Script Documentation**

**Overview**

This document provides an overview and explanation of a Python script created by Souza Paulo on 2024-05-29. The script reads data from a beer API and saves it into a historical folder as CSV files.

**Script Details**

**Creator Information**

* **Creator:** Souza Paulo
* **Creation Date:** 2024-05-29

**Description**

The script reads data from an API providing information about beers and breweries. It processes the data and saves it as CSV files in a specified historical folder for record-keeping and further analysis.

**Script Steps**

**Step 1: Import Main Libraries**

The script starts by importing the necessary libraries:

#Step 1: Import main libs to work

import pandas as pd

import requests

from datetime import datetime

import os

import warnings

warnings.filterwarnings("ignore")

 pandas: Used for data manipulation and storage.

 requests: Used for making HTTP requests to access the API.

 datetime: Used to generate timestamps for the filenames.

 os: Used for interacting with the operating system, specifically for file path operations.

 warnings: Used to suppress warnings for cleaner output.

### Step 2: Define the Function to Request and Save Data

The core of the script is a function that requests data from the API and saves it as a CSV file in a specified historical folder:

def save\_api\_response\_as\_csv(url, historicfolder):

    try:

        response = requests.get(url, verify=False)

        # If success, save the data into csv archive for historic

        if response.status\_code == 200:

            filename\_prefix = 'Beers'

            filename = '{}\_{}.csv'.format(filename\_prefix, datetime.now().strftime('%Y\_%m\_%d\_%H%M%S'))

            data = pd.DataFrame(response.json())

            data.to\_csv(os.path.join(historicfolder, filename), index=False)

            print("Data saved successfully as '{}'".format(filename))

            return True

        else:

            print("Error in request:", response.status\_code)

            return False

    except Exception as e:

        print("Error:", e)

        return False

#### Parameters:

* url (str): The API endpoint URL to request data from.
* historicfolder (str): The path to the folder where the CSV files will be saved.

#### Functionality:

1. **HTTP Request:** The function makes a GET request to the specified URL.
2. **Response Handling:** If the request is successful (status code 200), it processes the JSON data.
3. **Data Storage:** The data is converted to a pandas DataFrame and saved as a CSV file in the historical folder with a timestamped filename.
4. **Error Handling:** If the request fails or an exception occurs, it prints an error message and returns False.

### Execution

The function is called with the following parameters:

url = 'https://api.openbrewerydb.org/breweries'

historicfolder = r'C:\Users\paulo.souza\Desktop\Dados&Desenvolvimento\Aplicacao\Beers\HistoricFolder\RawData'

save\_api\_response\_as\_csv(url, historicfolder)

* **URL:** The API endpoint providing brewery data.
* **Historical Folder Path:** The directory where the CSV files will be stored.

**Overview**

This document provides an overview and explanation of a Python script created to manage SQL database tables and stored procedures related to beer data. The script checks for the existence of tables, creates them if necessary, and executes stored procedures to transform and load data.

**Script Details**

**Creator Information**

* **Creator:** Souza Paulo
* **Creation Date:** [Assume creation date if provided]

**Description**

The script establishes a connection to a SQL Server database, checks for the existence of specified tables, creates them if they do not exist, and defines stored procedures to process and transform data from a bronze layer to a silver layer and from a silver layer to a gold layer.

**Script Steps**

**Step 1: Import Main Libraries**

The script starts by importing the necessary libraries:

import pyodbc

import pandas as pd

### Step 2: Define SQL Connection Variables

The script defines the connection parameters required to connect to the SQL Server:

#Step 2:Create SQL connection variables

server = 'BeersServerTest'

username = 'BeerUser'

password = 'BeerUser'

driver = '{ODBC Driver 17 for SQL Server}'

### Step 3: Function to Create Table If Not Exists

A function is defined to check for the existence of a table and create it if it does not exist:

# Step 3: Função para criar tabela se não existir

def CreateTableIfNotExists(database,TableName, Schema):

    try:

        # Estabelece a conexão

        database = database

        # Cria a string de conexão

        connection\_string = f'DRIVER={driver};SERVER={server};PORT=1433;DATABASE={database};UID={username};PWD={password}'

        conn = pyodbc.connect(connection\_string)

        cursor = conn.cursor()

        # Verifica se a tabela existe

        check\_table\_query = f"""

        IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = '{TableName}')

        BEGIN

            {Schema}

        END

        """

        # Executa o comando

        cursor.execute(check\_table\_query)

        conn.commit()

        cursor.close()

        print(f"Tabela '{TableName}' verificada/criada com sucesso.")

    except Exception as e:

        print(f"Ocorreu um erro: {e}")

    finally:

        conn.close()

#### Parameters:

* database (str): The database name.
* TableName (str): The name of the table to check/create.
* Schema (str): The SQL schema for creating the table.

#### Functionality:

1. **Connection Setup:** Establishes a connection to the specified database.
2. **Table Existence Check:** Uses an SQL query to check if the table exists and creates it if not.
3. **Error Handling:** Catches and prints any exceptions that occur during execution.

### Step 4: Creating Tables

The script then uses the CreateTableIfNotExists function to create tables in different databases if they do not already exist:

# Create Bronze table

database = 'db\_bronze'

TableName = 'dbo.BeerBronzeLayer'

Schema = """

CREATE TABLE dbo.BeerBronzeLayer (

    IdRow INT PRIMARY KEY,

    Name NVARCHAR(255),

    BreweryType NVARCHAR(255),

    AddressOne NVARCHAR(255),

    AddressTwo NVARCHAR(255),

    AddressThree NVARCHAR(255),

    City NVARCHAR(255),

    StateProvince NVARCHAR(255),

    PostalCode NVARCHAR(255),

    Country NVARCHAR(255),

    Longitude NVARCHAR(255),

    Latitude NVARCHAR(255),

    Phone NVARCHAR(255),

    WebSiteURL NVARCHAR(255),

    State NVARCHAR(255),

    Street NVARCHAR(255),

    LineHash NVARCHAR(255),

    FileName NVARCHAR(255),

    BlockHash NVARCHAR(255),

    InsertDate DATETIME

)

"""

CreateTableIfNotExists(TableName, Schema)

# Create Silver table

database = 'db\_silver'

TableName = 'BeerSilverLayer'

Schema = """

CREATE TABLE dbo.BeerSilverLayer (

    IdRow INT PRIMARY KEY,

    Name NVARCHAR(255),

    BreweryType NVARCHAR(255),

    AddressOne NVARCHAR(255),

    AddressTwo NVARCHAR(255),

    AddressThree NVARCHAR(255),

    City NVARCHAR(255),

    StateProvince NVARCHAR(255),

    PostalCode NVARCHAR(255),

    Country NVARCHAR(255),

    Longitude NVARCHAR(255),

    Latitude NVARCHAR(255),

    Phone NVARCHAR(255),

    WebSiteURL NVARCHAR(255),

    State NVARCHAR(255),

    Street NVARCHAR(255),

    LineHash NVARCHAR(255),

    BlockHash NVARCHAR(255),

    InsertDate DATETIME

)

"""

CreateTableIfNotExists(TableName, Schema)

# Create Gold table

database = 'db\_gold'

TableName = 'dbo.BeerGoldLayer'

Schema = """

CREATE TABLE dbo.BeerGoldLayer (

    IdRow INT PRIMARY KEY,

    Name NVARCHAR(255),

    BreweryType NVARCHAR(255),

    AddressOne NVARCHAR(255),

    City NVARCHAR(255),

    State NVARCHAR(255),

    Street NVARCHAR(255),

    PostalCode NVARCHAR(255),

    Country NVARCHAR(255),

    Longitude NVARCHAR(255),

    Latitude NVARCHAR(255),

    Phone NVARCHAR(255),

    WebSiteURL NVARCHAR(255),

    LineHash NVARCHAR(255),

    BlockHash NVARCHAR(255),

    InsertDate DATETIME

)

"""

CreateTableIfNotExists(TableName, Schema)

### Step 5: Define and Execute Silver Procedure

A function is defined to create and execute a stored procedure to process data from the bronze layer to the silver layer:

def ExecuteSilverProcedure(Procedure, TargetTable):

    try:

        #Step 2:Create SQL connection variables

        server = 'BeersServerTest'

        database = 'db\_silver'

        username = 'BeerUser'

        password = 'BeerUser'

        driver = '{ODBC Driver 17 for SQL Server}'

        # Cria a string de conexão

        connection\_string = f'DRIVER={driver};SERVER={server};PORT=1433;DATABASE={database};UID={username};PWD={password}'

        # Estabelece a conexão

        conn = pyodbc.connect(connection\_string)

        # Ler a tabela de origem

        query = f"""

        CREATE {Procedure} AS

            BEGIN

                    DECLARE @InsertDatas TABLE (

                    BronzeLineHash VARCHAR(255),

                    SilverLineHash VARCHAR(255)

                );

                -- Preenchimento da tabela @InsertDatas com os resultados da consulta

                INSERT INTO @InsertDatas (BronzeLineHash, SilverLineHash)

                SELECT DISTINCT

                    Bronze.LineHash AS BronzeLineHash,

                    Silver.LineHash AS SilverLineHash

                FROM

                    db\_bronze.dbo.BeerBronzeLayer AS Bronze

                LEFT JOIN

                    db\_silver.dbo.BeerSilverLayer AS Silver

                ON

                    Bronze.LineHash = Silver.LineHash

                WHERE

                    Silver.LineHash IS NULL

                ;WITH s1 AS (

                    SELECT

                        Name,

                        BreweryType,

                        AddressOne,

                        AddressTwo,

                        AddressThree,

                        City,

                        StateProvince,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        State,

                        Street,

                        LineHash,

                        BlockHash,

                        CONVERT(DATETIME,InsertDate) AS InsertDate

                    FROM db\_bronze.dbo.BeerBronzeLayer Bronze

                    LEFT JOIN

                    (

                        SELECT DISTINCT

                            BronzeLineHash

                        FROM

                            @InsertDatas

                    ) AS InsertDatas

                    ON

                        Bronze.LineHash = InsertDatas.BronzeLineHash

                    WHERE InsertDatas.BronzeLineHash IS NOT NULL

                )

                INSERT INTO db\_silver.dbo.BeerSilverLayer (

                    -- Colunas

                        Name,

                        BreweryType,

                        AddressOne,

                        AddressTwo,

                        AddressThree,

                        City,

                        StateProvince,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        State,

                        Street,

                        LineHash,

                        BlockHash,

                        InsertDate

                )

                SELECT

                    -- Colunas

                        Name,

                        BreweryType,

                        AddressOne,

                        AddressTwo,

                        AddressThree,

                        City,

                        StateProvince,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        State,

                        Street,

                        LineHash,

                        BlockHash,

                        InsertDate

                FROM s1;

                WITH CTE AS (

                SELECT

                    \*,

                    ROW\_NUMBER() OVER (PARTITION BY LineHash ORDER BY InsertDate DESC)AS rn

                FROM

                    db\_silver.dbo.BeerSilverLayer WITH (NOLOCK))

                DELETE

                FROM

                    CTE

                WHERE rn > 1;

                UPDATE db\_silver.dbo.BeerSilverLayer

                SET

                    AddressOne = ISNULL(AddressOne, 'Others'),

                    AddressTwo = ISNULL(AddressTwo, 'Others'),

                    AddressThree = ISNULL(AddressThree, 'Others'),

                    Longitude = ISNULL(Longitude, '0'),

                    Latitude = ISNULL(Latitude, '0'),

                    Phone = ISNULL(Phone, 'Others'),

                    website\_url = ISNULL(website\_url, 'WithoutWebSite')

                WHERE

                    AddressOne IS NULL OR

                    AddressTwo IS NULL OR

                    AddressThree IS NULL OR

                    Longitude IS NULL OR

                    Latitude IS NULL OR

                    Phone IS NULL OR

                    website\_url IS NULL;

            END

        """

        # Inserir os dados transformados na tabela de destino

        cursor = conn.cursor()

        cursor.execute(query)

        conn.commit()

        cursor.close()

        print(f"Procedure criada para a tabela {TargetTable}")

    except Exception as e:

        print(f"Ocorreu um erro: {e}")

    finally:

        conn.close()

Procedure = 'dbo.sp\_SilverProcedure',

TargetTable = 'BeerSilverLayer'

ExecuteSilverProcedure(Procedure, TargetTable)

def ExecuteGoldProcedure(Procedure, TargetTable):

    try:

        #Step 2:Create SQL connection variables

        server = 'BeersServerTest'

        database = 'db\_gold'

        username = 'BeerUser'

        password = 'BeerUser'

        driver = '{ODBC Driver 17 for SQL Server}'

        # Cria a string de conexão

        connection\_string = f'DRIVER={driver};SERVER={server};PORT=1433;DATABASE={database};UID={username};PWD={password}'

        # Estabelece a conexão

        conn = pyodbc.connect(connection\_string)

        # Ler a tabela de origem

        query = f"""

        CREATE {Procedure} AS

            BEGIN

                    DECLARE @InsertDatas TABLE (

                    SilverLineHash VARCHAR(255),

                    GoldLineHash VARCHAR(255)

                );

                -- Preenchimento da tabela @InsertDatas com os resultados da consulta

                INSERT INTO @InsertDatas (SilverLineHash, GoldLineHash)

                SELECT DISTINCT

                    Silver.LineHash AS BronzeLineHash,

                    Gold.LineHash AS SilverLineHash

                FROM

                    db\_silver.dbo.BeerSilverLayer AS Silver

                LEFT JOIN

                    db\_gold.dbo.BeerGoldLayer AS Gold

                ON

                    Silver.LineHash = Gold.LineHash

                WHERE

                    Gold.LineHash IS NULL

                ;WITH s1 AS (

                    SELECT

                        Name,

                        BreweryType,

                        AddressOne,

                        AddressTwo,

                        AddressThree,

                        City,

                        StateProvince,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        State,

                        Street,

                        LineHash,

                        BlockHash,

                        InsertDate

                    FROM db\_silver.dbo.BeerSilverLayer Silver

                    LEFT JOIN

                    (

                        SELECT DISTINCT

                            SilverLineHash

                        FROM

                            @InsertDatas

                    ) AS InsertDatas

                    ON

                        Silver.LineHash = InsertDatas.SilverLineHash

                    WHERE InsertDatas.SilverLineHash IS NOT NULL

                )

                INSERT INTO db\_gold.dbo.BeerSilverLayer (

                    -- Colunas

                        Name,

                        BreweryType,

                        AddressOne,

                        City,

                        State,

                        Street,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        LineHash,

                        BlockHash,

                        InsertDate

                )

                SELECT

                    -- Colunas

                        Name,

                        BreweryType,

                        AddressOne,

                        City,

                        State,

                        Street,

                        PostalCode,

                        Country,

                        Longitude,

                        Latitude,

                        Phone,

                        WebSiteURL,

                        LineHash,

                        BlockHash,

                        InsertDate

                FROM s1;

                WITH CTE AS (

                SELECT

                    \*,

                    ROW\_NUMBER() OVER (PARTITION BY LineHash ORDER BY InsertDate DESC)AS rn

                FROM

                    db\_gold.dbo.BeerGoldLayer WITH (NOLOCK))

                DELETE

                FROM

                    CTE

                WHERE rn > 1;

            END

        """

        # Inserir os dados transformados na tabela de destino

        cursor = conn.cursor()

        cursor.execute(query)

        conn.commit()

        cursor.close()

        print(f"Procedure criada para a tabela {TargetTable}")

    except Exception as e:

        print(f"Ocorreu um erro: {e}")

    finally:

        conn.close()

Procedure = 'dbo.sp\_SilverProcedure',

TargetTable = 'BeerGoldLayer'

ExecuteSilverProcedure(Procedure, TargetTable)

#### Parameters:

* Procedure (str): The name of the stored procedure.
* TargetTable (str): The target table to process data into.