```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul  7 17:11:50 2021

@author: Lindsay Kelly
"""

import json
import pytz
import pandas as pd
import numpy as np
import datetime as dt
import os
import sys
file_dir = os.path.dirname(__file__)
sys.path.append(file_dir)
import geotab_testing as gps


cwd=os.getcwd()

utc = pytz.UTC
pst = pytz.timezone('America/Los_Angeles')

datetime_str_format = lambda d: pd.to_datetime(d).__format__('%Y-%m-%d %H:%M:%S')
datetime_dt_format = lambda d: pst.localize(dt.datetime.strptime(d, '%Y-%m-%d'))
datetime_dt_str = lambda d: dt.datetime.strftime(d,'%Y-%m-%d')

file_dir=lambda f, x: os.path.join(f,x)


####################################################

def save_json(df, params, file_name):
    """
    Parameters
    ----------
    df : dataframe to be saved to the project as a json file
    params : the configuration file (config.json) details to be used
    file_name : the output filename which will be saved.

    Returns: length of the dataframe saved to the json file
    """
    df.reset_index(drop=True)
    file_dir=os.path.join(params["folder_location"],params[file_name])
    df.to_json(file_dir,orient=params['json_orient'])
    print('number of trip records:',len(df))
    return len(df)

####################################################

def get_device_info_data(params, device_list, gps_conn=None):
    if params['use file']==True:

device_status_df=pd.read_json(file_dir(params["folder_location"],params['device_status_json']),orient
=params['json_orient'])
        num_rows=len(device_status_df)


    else:
        device_status_df=gps_conn.getDeviceInfo(device_list)
        # check the device status to confirm the vehicle was in use recently (there is a GPS trip
record within the search dates)
        device_status_df['include'] = np.where(device_status_df['last_gps_record']
<datetime_dt_format(params[params['time_period']]['end']),0,1)
        num_rows = save_json(device_status_df, params, "device_status_json")

    print('---------------------')
    print("device_status_df : device summary details ")
    print(f"device status data complete, {num_rows} records returned ")
    print('---------------------')
    return device_status_df
####################################################

def get_trip_summary_data(params, device_list, gps_conn=None):

    if params['use file']==True:

summary_df=pd.read_json(file_dir(params["folder_location"],params['trip_summary_json']),orient=params
['json_orient'])
        num_rows=len(summary_df)

    else:
        summary_df=gps.get_trip_summary_data(params, check_trips=False, gps_conn=gps_conn,
device_list=device_list)
        num_rows = save_json(summary_df, params, "trip_summary_json")

    print('---------------------')
    print("summary_df : vehicle trip summary statistics ")
    print(f"vehicle trip summary data complete, {num_rows} records returned ")
    print('---------------------')
    print(summary_df.head(2))
    return summary_df

####################################################

def get_trip_gps_data(params, df, gps_conn=None):

    if params['use file']==True:

trip_df=pd.read_json(file_dir(params["folder_location"],params['trip_gps_json']),orient=params['json_
orient'])
        num_rows=len(trip_df)
    else:
        trip_df=gps.get_trip_pts(params,gps_conn=gps_conn,df=df)
        num_rows = save_json(trip_df, params, "trip_gps_json")

    print('---------------------')
    print("trip_df : individual gps points for trips")
    print(f"vehicle trip detailed gps data complete, {num_rows} records returned ")
    print('---------------------')
    return trip_df

####################################################

def get_exception_gps_data(params, device_list=[], gps_conn=None):
    try:
        if params['use file']==True:

exception_df=pd.read_json(file_dir(params["folder_location"],params['exception_gps_json']),orient=par
ams['json_orient'])
            num_rows=len(exception_df)

        else:
            exception_df=gps.get_gps_exceptions(params,exception_item='street sweeper
engaged',device_list=device_list,gps_conn=gps_conn, df=None)
            num_rows = save_json(exception_df, params, "exception_gps_json")
    except:
        num_rows=0
        exception_df=None

    print('---------------------')
    print("exception_df : individual date time records of exception events - street sweeping
equipment usage")
    print(f"vehicle exception gps data complete, {num_rows} records returned ")
    print('---------------------')
    return exception_df

####################################################

def get_cov_bikelane_data(params):

    if params['use file']==True:

bikelane_df=pd.read_json(file_dir(params["folder_location"],params['cov_bikelanes_json']),orient=para
ms['json_orient'])
        num_rows=len(bikelane_df)
    else:
        open_data=gps.COV_open_data(params)
        bikelane_df=open_data.get_bike_lanes()
        num_rows = save_json(bikelane_df, params, "cov_bikelanes_json")

    print('---------------------')
    print("bikelane_df : City of Vancouver Open Data - Public Bike Lanes - GIS dataset")
    print(f"bike lane GIS data complete, {num_rows} records returned ")
    print('---------------------')
    return bikelane_df

####################################################

def get_cov_arterial_streets_data(params):
    if params['use file']==True:

arterial_streets_df=pd.read_json(file_dir(params["folder_location"],params['cov_arterials_json']),ori
ent=params['json_orient'])
        num_rows=len(arterial_streets_df)
    else:
        open_data=gps.COV_open_data(params)
        arterial_streets_df=open_data.get_street_segments()
        num_rows = save_json(arterial_streets_df, params, "cov_arterials_json")

    print('---------------------')
    print("arterial_streets_df : City of Vancouver Open Data - Arterial Core Traffic Streets - GIS
dataset")
    print(f"arterial street segement GIS data complete, {num_rows} records returned ")
    print('---------------------')
    return arterial_streets_df
```