```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon May 27 18:59:50 2019

@author: torjus

THIS PYTHON FILE IS THE CODE FOR IMPORTING AND PREPROCESSING DATA

"""

#DataImport and data preprocessing
import datetime
import sqlite3
import time
import pandas as pd
import unixTimeConvert as UTC
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile
from sqlalchemy import create_engine

def dataExtraction1(filepath):
    #Connect to SQLlite database
    conn = sqlite3.connect(filepath)

    #Defining SQL string
    SQLstring1 = "SELECT unixtime,mmsi,sog,nav_status,latitude,longitude \
        FROM %s ORDER BY UNIXTIME \
        ASC" % ('messagetype1')

    #Extract data from database
    with conn:
        df1 = pd.read_sql_query(SQLstring1,conn)
    conn.close()
    df1['DateTime'] = pd.to_datetime(df1['unixtime'],unit='s')
    df1.set_index('DateTime',inplace=True,drop=False)

    uniqueMMSI1 = df1['mmsi'].unique()
    return df1, uniqueMMSI1

def dataExtraction2(filepath):
    #Connect to SQLlite database
    conn = sqlite3.connect(filepath)

    #Defining SQL string
    SQLstring1 = "SELECT unixtime,mmsi,sog,nav_status,latitude,longitude \
        FROM %s ORDER BY UNIXTIME \
        ASC" % ('messagetype2')

    #Extract data from database
    with conn:
        df2 = pd.read_sql_query(SQLstring1,conn)
    conn.close()
    df2['DateTime'] = pd.to_datetime(df2['unixtime'],unit='s')
    df2.set_index('DateTime',inplace=True,drop=False)

    uniqueMMSI2 = df2['mmsi'].unique()
    return df2, uniqueMMSI2

def dataExtraction3(filepath):
    #Connect to SQLlite database
    conn = sqlite3.connect(filepath)

    #Defining SQL string
    SQLstring1 = "SELECT unixtime,mmsi,sog,nav_status,latitude,longitude \
        FROM %s ORDER BY UNIXTIME \
```

```
        ASC" % ('messagetype3')

    #Extract data from database
    with conn:
        df3 = pd.read_sql_query(SQLstring1,conn)
    conn.close()
    df3['DateTime'] = pd.to_datetime(df3['unixtime'],unit='s')
    df3.set_index('DateTime',inplace=True,drop=False)

    uniqueMMSI3 = df3['mmsi'].unique()
    return df3, uniqueMMSI3

def dataExtraction5(filepath):
    #Connect to SQLlite database
    conn = sqlite3.connect(filepath)

    #Defining SQL string
    SQLstring1 = "SELECT unixtime,mmsi,imo_num,callsign,name,shiptype,\
        destination,length,breadth,eta_month,eta_day,eta_hour,eta_minute, \
        draught FROM %s ORDER BY UNIXTIME \
        ASC" % ('messagetype5')

    #Extract data from database
    with conn:
        df5 = pd.read_sql_query(SQLstring1,conn)
    conn.close()
    df5['DateTime'] = pd.to_datetime(df5['unixtime'],unit='s')
    df5.set_index('DateTime',inplace=True,drop=False)

    uniqueMMSI5 = df5['mmsi'].unique()
    imoAll = df5['imo_num'].unique()
    return df5, uniqueMMSI5, imoAll

def removeErrorData(df):
    df_out = df[(df['sog']<30) & (df['sog']>=0)]
    df_out = df_out[(df_out['latitude']<=90) & (df_out['latitude']>=-90) &
                    (df_out['longitude']<=180) & (df_out['longitude']>=-180)]
    df_out = df_out.copy()
    uniqueMMSI = df_out['mmsi'].unique()
    return df_out, uniqueMMSI

def mergeDynamic(df1,df2,df3):
    frames = [df1,df2,df3]
    df_merge = pd.concat(frames)
    df=df_merge.copy()
    df.sort_index(inplace=True)
    uniqueMMSI = df['mmsi'].unique()
    return df, uniqueMMSI


def addVesselinformation(df_final,df_V,uniqueMMSI):
    #VesselCategories: Small Carrier =1, Conventional = 2,..
    #New Panamax = 3, QflexOrQmax = 3
    #Age Categories: Age(11+) = 1, Age(6-10) = 2, Age(0-5) = 3
    start = time.time()
    df = df_final.copy()


    df['gasCap'] = 0
    df['category']=0
    df['serviceSpeed']=0
    df['age']=0
    df_out = pd.DataFrame(columns = df.columns)
    for mmsi in uniqueMMSI:
        rowFrame = df_V[df_V['mmsi']==mmsi]
        df_temp = df[df['mmsi']==mmsi]
        if len(rowFrame)==0:
            continue
        row = rowFrame.iloc[0]
        gasCap = row['gasCap']
        category = row['category']
        serviceSpeed = row['serviceSpeed']
        ageGroup = row['ageGroup']
```

```
        df_temp['gasCap'] = gasCap
        df_temp['category']=category
        df_temp['serviceSpeed']=serviceSpeed
        df_temp['ageGroup']=ageGroup

        df_out= df_out.append(df_temp)

    df_out.sort_index(inplace=True)
    df_no_info = df_out[df_out['gasCap']==0]
    end = time.time()
    tot = end-start
    return df_out,df_no_info

def reduceMessageFreq(df1,uniqueMMSI1):

    start = time.time()

    df_out = pd.DataFrame(columns = df1.columns)
    secMin = 60 #seconds
    for mmsi in uniqueMMSI1:
        df_temp = df1[df1['mmsi'] == mmsi]
        if len(df_temp)==0:
            continue
        unixLast = df_temp['unixtime'].iloc[0]
        delete = [0]*len(df_temp)
        delete[0]=1 #Add first message automatic
        df_temp.reset_index(drop=True,inplace=True)
        for i, row in enumerate(df_temp.itertuples()):
            if (row.unixtime-unixLast) > secMin:
                delete[i]=1
                unixLast=row.unixtime
        df_temp['delete'] = delete
        df_temp = df_temp[df_temp['delete']==1]

        df_out = df_out.append(df_temp)

    df_out.set_index('DateTime',inplace=True,drop=False)
    df_out.sort_index(inplace=True)
    end = time.time()
    tot = end-start

    return df_out,tot


def makeNewSqlite(df,engine_name):
    engine = create_engine(engine_name, echo=False)
    df.to_sql('messagetype1', con=engine,if_exists='replace')
    #engine.execute("SELECT * FROM messagetype1").fetchall()
    engine.dispose
```

```
def LNGTerminals():
    df = pd.read_excel('LNGRecievingTerminals.xlsx',sheet_name='ForPython')
    return df
def LNGProduction():
    df = pd.read_excel('LiquifactionPlants.xlsx',sheet_name='ForPython')
    return df
def importVesselData():
    df_V = pd.read_excel('vSheet_final.xlsx',sheet_name='final')
    return df_V

def createFinalVesselSheet(df_final,df_V,uniqueMMSI):
    #VesselCategories: Small Carrier =1, Conventional = 2,..
    #New Panamax = 3, QflexOrQmax = 4
    #Age Categories: Age(11+) = 1, Age(6-10) = 2, Age(0-5) = 3

    vSheet = pd.DataFrame(columns=['imo_num','mmsi','category','year','ageGroup','serviceSpeed','gasCap'])
    #vSheet['mmsi']= uniqueMMSI

    for i,mmsi1 in enumerate(uniqueMMSI):
        df_temp = df_V[df_V['mmsi']==mmsi1]
        if(len(df_temp)==0):
            vSerie = pd.Series([0,mmsi1,0,0,0,0,0], index=vSheet.columns)
            vSheet = vSheet.append(vSerie,ignore_index=True)
        else:
```

```
            vSerie = df_V[df_V['mmsi']==mmsi1].iloc[0]
            vSheet = vSheet.append(vSerie,ignore_index=True)

    vNotFound = vSheet[vSheet['gasCap']==0]
    vSheet.to_excel('vSheet.xlsx')
    return vSheet,vNotFound
```