```
# -*- coding: utf-8 -*-
print('Subfleeting tool - version 6/14/2022 - V7')
print('Preparing inputs...')

#1.11
#HPTS can be taken from file
#All Other reseted always
#HPT, HPTS, HPTB Weibulls used for B50 calculation and define date of next resetting event
#HPTS, HPTB Weibulls according to available hardware based on SV date
#HPT Weibull based on HPTB configuration
#All progresion till xx9 subfleet


#1.10
#HPT Progression till 5

#1.09
#Entry into service update
#looup sheet update - new structed for HPTS1S configuration, new logic
#durability Weibull - detailed (all thrusts) model for CDN and HPT
#developed methodology for mature configuration availability (HPTS1S / HPTS1B)
#infoormation about No of resetting events for configuration >Phase 1 limitated to "1" - shrouds

#1.08
#TOW Cap logic developed - SV number taken into consideration, progression logic added

#1.07
#performance updates

#1.06
#TOW CAP bucket / Weibul calculation (/2) / subfleeting added
#"progression unavailable" option in Workscope Failure sheet in case that particuar case is not predicted by model

#1.05
# Shround Weibull Safran file removed from inputs (all information taken from Technical assumption file)
# correction of eW / dW / qW / subfleets frmaes (new Technica assumption file)
# new rules for subfleet assigment (removal o 500 / Shoruds configuration assigment based on region / 1A/1B differences)

#1.04
```

```
#Configuration

#1.03
#Retainer Weibull out

#1.02
#additional PN and names for mature configurations
#shrouds: phase 4 and phase 5 = Mature, phase 2 = phase 3 the same Weibull (for phase 2),
# Weibull assigned based on configuration not subfleet
#LPTR: Mature replaced by LPTRR_Redesign and LPTRR_Redesign_Group
#HPTS1B: Mature replaced by Blade_Redesign and Blade_Redesign_Group

#1.01
#cut-off date is taken from FSA output
#svc data base can be uploaded as in CSV or xlx format


#------------------------------libraries-----------------------------------------------------
import pandas as pd
import datetime
import numpy as np
import os
import time
#------------------------------working directory-----------------------------------------------
start_time = time.time()

cwd=os.getcwd()
os.chdir('C:/Users/502340035/Desktop/tool/2022')
#os.listdir('C:/Users/502340035/Desktop/tool/2022')
#------------------------------input files/dates-----------------------------------------------
time_before = time.time()

#svc = 'All_Report_04222022_LEAP-1A.csv'
#svc = 'All_Report_04222022_LEAP-1B.csv'
#svc = 'All_Report_1A.csv'
#svc = 'All_Report_1A (1).csv'
#svc = 'LEAP-1B.csv'
#svc = pd.ExcelFile('All_Report_1A (1).csv')
#svc = 'All_Report_1A.csv'
```

```
#svc = 'LEAP-1A.csv'
svc = 'LEAP-1B (1).csv'


#fsa = 'FSA LEAP-1A AIR INDIA LEAP-1A 05-31-2022.xlsx'
#fsa = 'FSA LEAP-1A PEGASUS LEAP-1A 05-31-2022.xlsx'
#fsa = pd.ExcelFile('FSA LEAP-1A PEGASUS LEAP-1A 05-31-2022.xlsx')
#fsa = pd.ExcelFile('FSA LEAP-1A GULF AIR LEAP-1A 06-30-2022.xlsx')
#fsa = pd.ExcelFile('FSA LEAP-1B FLYDUBAI LEAP-1B 06-30-2022.xlsx')
#fsa = pd.ExcelFile('FSA LEAP-1A GULF AIR LEAP-1A 06-30-2022A26.xlsx')
#fsa = pd.ExcelFile('FSA_update Gulf A33.xlsx')
fsa = pd.ExcelFile('FSA LEAP-1B FLYDUBAI LEAP-1B 06-30-2022 (2).xlsx')

#model = pd.ExcelFile('LEAP_Technical_Assumption_Summary_REV6_2021-08-13.v1.xlsm')
#model = pd.ExcelFile(r'C:\Users\502340035\Desktop\tool\2022\LEAP_Technical_Assumption_Summary_Rev7 _6_20 with CDN adn EGT.xlsm')
#model = pd.ExcelFile(r'C:\Users\502340035\Desktop\tool\2022\LEAP_Technical_Assumption_Summary_Rev7 _6_20  ALL.xlsm')
#model = pd.ExcelFile('LEAP_Techniçal_Assumption_Summary_Rev7 _6_20  final.xlsm')
model = pd.ExcelFile('LEAP_Technical_Assumption_Summary_Rev7 _6_22 FINAL.xlsm')

#utlization and FL for threshold calculations
#HRs_ut = 3800
#FL = 3
#A33
HRs_ut = 5000
FL = 2.2
sev1 = 0.377 #LEAP_2015_Hot_Section_Severity_With_Trim
sev2 = 0.873 #CFM56 2013 Thrust Severity A
TAT = 120
#A26
#HRs_ut = 4200
#FL = 1.42
#sev1 = 0.5263 #LEAP_2015_Hot_Section_Severity_With_Trim
#sev2 = 1.5022 #CFM56 2013 Thrust Severity A
#TAT = 120


#severity_factor_HPT_section = CDN / HPT / HPTB = 0.36 harsh/ 0.9
#severity_thrust_severity_A shrouds = 0.83 harsh / 0.9
```

```
print("Input loaded in {} seconds. Total time {} seconds".format(round(time.time()-time_before,1), round(time.time()-start_time,1)))


#-----------------------------functions-----------------------------------------------------
time_before = time.time()

def clock (ESN_f, bucket_f, time):
    SVdata = 0
    Modata = 0
    QTdata = 0
    SVdata = cmr_cup[time].loc[(cmr_cup['Event Type'].astype(str) == 'SV') & (cmr_cup['ESN'] == ESN_f)].max()
    if SVdata!=SVdata: SVdata=0  # NaN elimination

    if bucket_f == 'Other':
        Modata = cmr_cup[time].loc[(cmr_cup['Event Type'].astype(str) == 'Module SV') & (cmr_cup['ESN'] == ESN_f)].max()
    if bucket_f != 'Other':
        Modata = cmr_cup[time].loc[(cmr_cup['Event Type'].astype(str) == 'Module SV') &\
                        (cmr_cup['ESN'] == ESN_f) & (cmr_cup['FM Weibull Bucket'] == bucket_f)].max()
    if Modata!=Modata: Modata=0  # NaN elimination
    if bucket_f == 'HPT_S1_Blade':
        QTdata = cmr_cup[time].loc[(cmr_cup['ESN'] == ESN_f) & (cmr_cup['Wrkscp.HPT_S1_Blade'] == 3)].max()
    if bucket_f == 'HPT_S1_Shroud':
        QTdata = cmr_cup[time].loc[(cmr_cup['ESN'] == ESN_f) & (cmr_cup['Wrkscp.HPT_S1_Shroud'] == 3)].max()
    if QTdata!=QTdata : QTdata=0  # NaN elimination
    return Engine.iloc[Engine.index[Engine['Engine Serial Number'] ==\
                                    ESN_f].tolist()[0],Engine.columns.get_loc(time)] - max([SVdata, Modata, QTdata])

def counter (ESN_f, bucket_f):
    SVdata_c = 0
    Modata_c = 0
    QTdata_c = 0
    SVdata_c = cmr_cup['Event Number'].loc[(cmr_cup['Event Type'].astype(str) == 'SV') & (cmr_cup['ESN'] == ESN_f)].count()
    Modata_c = data['Event Number'].loc[(data['Event Type'].astype(str) == 'Module SV') &\
                   (data['ESN'] == ESN_f) & (data['FM Weibull Bucket'] == bucket_f)].count()

    if bucket_f == 'Other':
        return cmr_cup['Event Number'].loc[(cmr_cup['ESN'] == ESN_f) &\
                       (cmr_cup['Event Type'].isin(['QT','SV','Module SV']))].count()
```

```
    if bucket_f == 'HPT_S1_Blade':
        QTdata_c = cmr_cup['Event Number'].loc[((cmr_cup['ESN'] == ESN_f) &\
                           (cmr_cup['Wrkscp.HPT_S1_Blade'] == 3) & (cmr_cup['Event Type'].astype(str) != 'SV')) |\
                           ((cmr_cup['ESN'] == ESN_f) & (cmr_cup['Wrkscp.HPT_S1_Blade'] == 3) &\
                            (cmr_cup['Event Type'].astype(str) == 'Module SV') & (data['FM Weibull Bucket'] != bucket_f))].count()

    if bucket_f == 'HPT_S1_Shroud':
        QTdata_c = cmr_cup['Event Number'].loc[((cmr_cup['ESN'] == ESN_f) & (cmr_cup['Wrkscp.HPT_S1_Shroud'] == 3) &\
                           (cmr_cup['Event Type'].astype(str) != 'SV')) |\
                           ((cmr_cup['ESN'] == ESN_f) & (cmr_cup['Wrkscp.HPT_S1_Shroud'] == 3) &\
                            (cmr_cup['Event Type'].astype(str) == 'Module SV') & (cmr_cup['FM Weibull Bucket'] != bucket_f))].count()

    return  sum([SVdata_c, Modata_c, QTdata_c])


def hundreds (ESN_f, thrust_f):

    if thrust_f == 'LEAP-1A32': subfleets = subfleets1A
    if thrust_f == 'LEAP-1B28': subfleets = subfleets1B
    if thrust_f == 'LEAP-1C30': subfleets = subfleets1C

    try:
        conf = start_conf.loc[start_conf['ESN']==ESN_f].loc[start_conf['Event Date']==\
                              min(start_conf['Event Date'].loc[start_conf['ESN']==ESN_f])].iloc[0]
        if HPT_conf_round(conf['Conf_in.HPT Stage 1 Blade Group']) == 'Block': return 100
        if HPT_conf_round(conf['Conf_in.HPT Stage 1 Blade Group']) == 'Turkey':
            if conf['Conf_in.HPT Stage 1 Shroud Group'] == 'Phase 0': return 200
            if conf['Conf_in.HPT Stage 1 Shroud Group'] == 'Phase 1': return 300
            if conf['Conf_in.HPT Stage 1 Shroud Group'] in ['Phase 2','Phase 3']: return 400
            if conf['Conf_in.HPT Stage 1 Shroud Group'] == 'Phase 4': return 500

    except:
        start = data['Initial Engine Delivery Date'].loc[(data['ESN'] == ESN_f) & (data['Event Type'] == 'CES')]
        #if len(start)>0: delivery = datetime.datetime.strptime(start.iloc[0], '%Y-%m-%d').date()
        if len(start)>0: delivery = datetime.datetime.strptime(start.iloc[0], '%m/%d/%Y').date()
        if len(start)==0: delivery = datetime.datetime.strptime(Engine['Delivery Date']\
              .loc[Engine['Engine Serial Number'] == ESN_f].iloc[0],'%m/%d/%Y').date()

        if delivery >= subfleets['customer delivery'].iloc[-1].date():return 900
```

```
        for i in subfleets.index:
            if delivery < subfleets['customer delivery'][i].date() : return subfleets['ICAM ID'][i-1]


def counter_round (counter_f):
    if counter_f == 0: return 0
    if counter_f > 0: return 1

def Shroud_conf_round (config_f):
        if config_f == 'Phase 0': return 0
        if config_f == 'Phase 1': return 1
        else: return 2

def shroud_conf (subfleet_f, region_f, thrust_f):

    if thrust_f == 'LEAP-1A32':
        if subfleet_f in [100,200]: return  'Phase 0', '2789M01P01'
        if subfleet_f == 300: return  'Phase 1', '2796M21P01'
        if subfleet_f == 400: return  'Phase 3', '2796M21P02'
        if subfleet_f in [500,600,700]: return  'Phase 4', '2796M21P04'
        if region_f in ['Neutral','China']:
            if subfleet_f in [800,900]: return  'Phase 4', '2796M21P04'
        if region_f in ['India','Middle East']:
            if subfleet_f in [800,900]: return  'Phase 5', '2796M21P05'

    if thrust_f == 'LEAP-1B28':
        if subfleet_f in [100,200]: return  'Phase 0', '2653M19P02'
        if subfleet_f == 300: return  'Phase 1', '2790M11P01'
        if subfleet_f == 400: return  'Phase 3', '2790M11P02'
        if subfleet_f in [500,600,800,900]: return  'Phase 4', '2796M21P04'
        if subfleet_f == 900: return  'Phase 5', '2796M21P05'


    if thrust_f =='LEAP-1C30':
        if subfleet_f == 600: return  'Phase 5', '2747M92P05'
        if subfleet_f in [800,900]: return  'Phase 5', '2790M11P05'


def QT_counter_mix ():
```

```
    rows5=[]
    for ESN in ESN_list:
        if ESN in HPTS_P2_list:
#           row5 = [ESN, hundreds (ESN, thrust1), counter(ESN,'HPT_S1_Shroud'), 'P2']
            row5 = [ESN, hundreds (ESN, thrust1), counter_round(counter(ESN,'HPT_S1_Shroud')),
                    Hardware_Configuration['Conf_in.HPT Stage 1 Shroud Group'].loc[Hardware_Configuration['ESN'] == ESN].iloc[0]]
            rows5.append(row5)



        if ESN in HPT_P0P1_NotModified_list:
            row5 = [ESN, hundreds (ESN, thrust1), counter(ESN,'HPT_S1_Shroud'),
                    'Phase '+str(Shroud_conf_round(Hardware_Configuration['Conf_in.HPT Stage 1 Shroud Group'].\
                                                   loc[Hardware_Configuration['ESN'] == ESN].iloc[0]))]
            rows5.append(row5)
        if ESN in HPT_P0P1_Modified_list:
            row5 = [ESN, hundreds (ESN, thrust1), counter(ESN,'HPT_S1_Shroud'), label(ESN)]
            rows5.append(row5)

    ESN_conf = pd.DataFrame(rows5, columns=['ESN','subfleets','counter','configuration'])
    Conf_agr = pd.DataFrame(rows5, columns=['ESN','subfleets','counter','configuration']).\
    groupby(['subfleets','counter','configuration']).size().reset_index().rename(columns={0:'count'})
    return ESN_conf, Conf_agr


def label (ESN_f):
    label = ''
    for i in HPTS_up_list.loc[HPTS_up_list['ESN']==ESN_f].index:
        label = label + '-P' + str(Shroud_conf_round(HPTS_up_list['Conf_in.HPT Stage 1 Shroud Group'][i]))
        last_conf = str(Shroud_conf_round(HPTS_up_list['Conf_out.HPT Stage 1 Shroud Group'][i]))
    label = label+ '-P' + last_conf
    if label[0] == '-': label = label[1:]
#    if label.count('P') > 3: label = label[-8:]       #if label.count('P') > 3: label = label[:8] - altirbnatively
#    if label[-5:] == 'P0-P0': label = label[-5:]
    return label


def QT_counter_mix2 ():
    rows8=[]
```

```
    for ESN in ESN_list:
        row8 = [ESN, hundreds (ESN, thrust1), counter_round(counter(ESN,'HPT_S1_Blade')),
                HPT_conf_round(Hardware_Configuration['Conf_in.HPT Stage 1 Blade Group'].\
                               loc[Hardware_Configuration['ESN'] == ESN].iloc[0])]
        rows8.append(row8)

    ESN_conf = pd.DataFrame(rows8, columns=['ESN','subfleets','counter','configuration'])
    index=pd.DataFrame(rows8, columns=['ESN','subfleets','counter','configuration']).\
    groupby(['subfleets','counter','configuration']).size().reset_index().rename(columns={0:'count'})

    rows9=[0]
    tens = 0
    for i in range(1,len(index)):
        if index['subfleets'][i]==index['subfleets'][i-1]: tens = tens+1
        else: tens = 0
        rows9.append(tens)
    index['tens']=rows9
    return ESN_conf, index



def HPT_conf_round (configuration_f):
    if configuration_f == 'Blade_Redesign_Group': return 'Blade_Redesign_Group'
    if configuration_f == 'Turkey': return 'Turkey'
    else: return 'Block'


def dozens (ESN_f, bucket_f):
    if bucket_f not in ['HPT_S1_Blade','HPT_S1_Shroud']: return 0
    if bucket_f =='HPT_S1_Blade':
        t = HTPS1B_QT_conf_mix
        return 10 * t[1]['tens'].loc[(t[1]['configuration']==t[0]['configuration'].loc[t[0]['ESN']==ESN_f].iloc[0]) &\
                     (t[1]['counter']==t[0]['counter'].loc[t[0]['ESN']==ESN_f].iloc[0]) &\
                     (t[1]['subfleets']==t[0]['subfleets'].loc[t[0]['ESN']==ESN_f].iloc[0])].iloc[0]
    if bucket_f =='HPT_S1_Shroud':
        t = Shroud_QT_conf_mix
        return 10 *  t[1]['tens'].loc[(t[1]['configuration']==t[0]['configuration'].loc[t[0]['ESN']==ESN_f].iloc[0]) &\
                     (t[1]['counter']==t[0]['counter'].loc[t[0]['ESN']==ESN_f].iloc[0]) &\
```

```
                    (t[1]['subfleets']==t[0]['subfleets'].loc[t[0]['ESN']==ESN_f].iloc[0]) ].iloc[0]


def factor_durability (thrust_f, subfleet_f, region_f, bucket_f, parameter_f):

    if bucket_f =='FAN': cause ='Fan Module'
    if bucket_f =='HPC': cause ='HPC'
#    if bucket_f =='CDN': cause ='CDN'
    if bucket_f =='LPTN': cause ='LPTN1 / TCF'
    if bucket_f =='LPT': cause ='LPT Module'
#    if bucket_f =='Other': cause ='Other'
    if bucket_f =='Bearing': cause ='Bearings'
    if bucket_f =='High_Cost': cause ='High Cost'
    if bucket_f =='FOD': cause ='FOD'

#    if (bucket_f == 'CDN') and ('LEAP-1B28' in thrust_f): thrust_f = 'LEAP-1B28'

    try:
        if subfleet_f%10 == 1: SV_no=['01']
        if subfleet_f%10 != 1: SV_no=['02+']
        return dW[parameter_f].loc[(dW['Engine Serie'] == thrust_f) &\
                (dW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                (dW['Environment'] == region_f) & (dW['Cause'] == cause) & (dW['SV#'].isin(SV_no))].iloc[0]
    except: print(thrust_f, subfleet_f, region, cause)


def factor_other (thrust_f, subfleet_f, region_f, date_f, parameter_f):

    if subfleet_f%10 == 1: SV_no='01'
    if subfleet_f%10 != 1:
        SV_no='02+'
        if date_f >= pd.to_datetime('10/1/2023', format='%m/%d/%Y') and thrust_f == 'LEAP-1A32': subfleet_f = 900
        if date_f >= pd.to_datetime('8/1/2024', format='%m/%d/%Y') and thrust_f == 'LEAP-1B28': subfleet_f = 900
        if date_f >= pd.to_datetime('4/1/2025', format='%m/%d/%Y') and thrust_f == 'LEAP-1C30': subfleet_f = 900

    return dW[parameter_f].loc[(dW['Engine Serie'] == thrust_f) &\
            (dW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) & (dW['Environment'] == region_f) &\
            (dW['Cause'] == 'Other') & (dW['SV#']==SV_no)].iloc[0]
```

```
def factor_cdn (ESN_f, thrust_f, subfleet_f, region_f, date_f, parameter_f):

#        Eta4 = factor_cdn (thrust2, subfleet_CDN +i, region, date_wo_TAT, 'Eta')
#        Beta4 = factor_cdn (thrust2, subfleet_CDN +i, region, date_wo_TAT, 'Beta')
    if parameter_f == 'Eta': col = 'Scale (cycles)'
    if parameter_f == 'Beta': col = 'Shape'

    try:
        if thrust1 == 'LEAP-1A32':
            if subfleet_f%10 == 1:
                return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                           (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                           (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
            if subfleet_f%10 != 1:
                if date_f < subfleets1A['customer delivery'].loc[subfleets1A['ICAM ID'] == 600].iloc[0]:
                    return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                               (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                               (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')].iloc[0]
                if date_f >= subfleets1A['customer delivery'].loc[subfleets1A['ICAM ID'] == 600].iloc[0]:
                    if date_f > subfleets1A['customer delivery'].iloc[-1].date(): CDN_subfleet = 900
                    else:
                        for i in subfleets1A.index:
                            if date_f < subfleets1A['customer delivery'][i].date():
                                CDN_subfleet = subfleets1A['ICAM ID'][i-1]
                                break
                    return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Environment'] == region_f) &\
                               (cdnW['SV#'] == '2+') & (cdnW['Subfleets'].str.contains(str(CDN_subfleet)))].iloc[0]

        if thrust1 == 'LEAP-1B28':
            if subfleet_f%10 == 1:
                return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                           (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                           (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
            if subfleet_f%10 != 1:

                if region_f == 'Middle East': k=0.85
                else: k=0.75
```

```
k1=1

if (subfleet_f//100)*100 <= 600:

    if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='600') & (cdnW['Environment']==region_f) &\
                    (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].min():
        if (subfleet_f//100)*100 < 600:
            return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                        (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                        (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')].iloc[0]
        if (subfleet_f//100)*100 == 600:
            if parameter_f == 'Beta': return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                        (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                        (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
            if parameter_f == 'Eta': return k * cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                        (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                        (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
    else:

        if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='600') & (cdnW['Environment']==region_f) &\
                        (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max():
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='600') &\
                                 (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            temporary = temporary.sort_values(by='Date', ascending = True).reset_index(drop=True)
            for i in range(len(temporary)):
                if temporary['Date'].iloc[i] > date_f: return temporary[col][i-1]

        if date_f >= cdnW['Date'].loc[(cdnW['Subfleets']=='600') & (cdnW['Environment']==region_f) &\
                         (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max() and\
        CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==False:
            CDN_kit_upgrade.loc[CDN_kit_upgrade['ESN']==ESN_f,'flag'] = True
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='600') &\
                                 (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            return temporary[col].loc[temporary['Date'].idxmax()]

        if CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==True:
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='600') &\
                                 (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            if parameter_f == 'Beta': return temporary[col].loc[temporary['Date'].idxmax()]
```

```
            if parameter_f == 'Eta': return k1 * temporary[col].loc[temporary['Date'].idxmax()]

if (subfleet_f//100)*100 == 800:

    if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='800') & (cdnW['Environment']==region_f) &\
                    (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].min():
            if parameter_f == 'Beta': return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                        (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                        (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
            if parameter_f == 'Eta': return k * cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                        (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                        (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]

    else:
        if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='900') & (cdnW['Environment']==region_f) &\
                        (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max():
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets'].isin(['800','900'])) &\
                            (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            temporary = temporary.sort_values(by='Date', ascending = True).reset_index(drop=True)
            for i in range(len(temporary)):
                if temporary['Date'].iloc[i] > date_f: return temporary[col][i-1]

        if date_f >= cdnW['Date'].loc[(cdnW['Subfleets']=='900') & (cdnW['Environment']==region_f) &\
                        (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max() and\
                        CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==False:
            CDN_kit_upgrade.loc[CDN_kit_upgrade['ESN']==ESN_f,'flag'] = True
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='900') &\
                                (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            return temporary[col].loc[temporary['Date'].idxmax()]

        if CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==True:
            temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='900') &\
                                (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
            if parameter_f == 'Beta': return temporary[col].loc[temporary['Date'].idxmax()]
            if parameter_f == 'Eta': return k1 * temporary[col].loc[temporary['Date'].idxmax()]

if (subfleet_f//100)*100 == 900:

    if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='900') & (cdnW['Environment']==region_f) &\
```

```
                                (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].min():
                        if parameter_f == 'Beta': return cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                                    (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                                    (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]
                        if parameter_f == 'Eta': return k * cdnW[col].loc[(cdnW['Engine Serie'] == thrust_f) &\
                                    (cdnW['Subfleets'].str.contains(str((subfleet_f//100)*100),na=False)) &\
                                    (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '1')].iloc[0]

                else:
                    if date_f < cdnW['Date'].loc[(cdnW['Subfleets']=='900') & (cdnW['Environment']==region_f) &\
                                    (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max():
                        temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets'] == '900') &\
                                            (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
                        temporary = temporary.sort_values(by='Date', ascending = True).reset_index(drop=True)
                        for i in range(len(temporary)):
                            if temporary['Date'].iloc[i] > date_f: return temporary[col][i-1]

                    if date_f >= cdnW['Date'].loc[(cdnW['Subfleets']=='900') & (cdnW['Environment']==region_f) &\
                                    (cdnW['Engine Serie']==thrust_f) & (cdnW['SV#']=='2+')].max() and\
                                    CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==False:
                        CDN_kit_upgrade.loc[CDN_kit_upgrade['ESN']==ESN_f,'flag'] = True
                        temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='900') &\
                                            (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
                        return temporary[col].loc[temporary['Date'].idxmax()]

                    if CDN_kit_upgrade['flag'].loc[CDN_kit_upgrade['ESN']==ESN_f].iloc[0]==True:
                        temporary = cdnW.loc[(cdnW['Engine Serie'] == thrust_f) & (cdnW['Subfleets']=='900') &\
                                            (cdnW['Environment'] == region_f) & (cdnW['SV#'] == '2+')]
                        if parameter_f == 'Beta': return temporary[col].loc[temporary['Date'].idxmax()]
                        if parameter_f == 'Eta': return k1 * temporary[col].loc[temporary['Date'].idxmax()]
    except: print(ESN_f, thrust_f, subfleet_f, region_f, date_f, parameter_f)
```

---

```
def factor_egt (thrust_f, subfleet_f, region_f, parameter_f):
    if subfleet_f%10 == 1: SV_no='01'
    if subfleet_f%10 != 1: SV_no='02+'

    if region_f in ['India','Middle East']: reg = 'India, Middle East'
    if region_f in ['Neutral','China']: reg = 'Neutral, China'
```

```
    return eW[parameter_f].loc[(eW['Engine Serie'] == thrust_f) & (eW['SV#'] == SV_no) &\
            (eW['Environment'] == reg) & (eW['Subfleet'].str.contains(str((subfleet_f//100)*100),na=False))].iloc[0]


def factor_cap (thrust_f, parameter_f, subfleet_f):
    if subfleet_f%10 == 1: SV_no= 1
    if subfleet_f%10 != 1: SV_no='2+'
    return cW[parameter_f].loc[(cW['engine_serie'] == thrust_f) & (cW['sv_num'] == SV_no)].iloc[0]


def HPTS1B_conf (subfleet_f,region_f, thrust_f):

    if thrust_f == 'LEAP-1A32':
        if subfleet_f == 100: return 'Block 3-4', '2553M91G08'
        if subfleet_f in  [200,300,400,500,600]: return 'Turkey', '2747M92P01'
        if subfleet_f > 600: return 'Blade_Redesign_Group', '2825M11G01'
    if thrust_f == 'LEAP-1B28':
        if subfleet_f == 100: return 'Block 3-4', '2010M42P06'
        if subfleet_f in  [200,300,400,500,600]: return 'Turkey', '2747M52P02'
#        if subfleet_f == 700:
#            if region_f in ['Neutral','China']: return 'Blade_Redesign_Group', '2929M96P01'
#            if region_f in ['India','Middle East']: return 'Turkey', '2747M52P02'
        if subfleet_f > 600: return 'Blade_Redesign_Group', '2929M96P01'
    if thrust_f =='LEAP-1C30':
        if subfleet_f == 600: return 'Turkey', '2747M92P01'
        if subfleet_f > 600: return 'Blade_Redesign_Group', 'Blade_Redesign'

def MHPTS1B_start (thrust_f):
    if thrust_f =='LEAP-1A32':
        return datetime.datetime.fromisoformat(str(subfleets1A['customer delivery'].\
                loc[subfleets1A['Blade']=='2825M11G01'].iloc[0]))
    if thrust_f =='LEAP-1B28':
        return datetime.datetime.fromisoformat(str(subfleets1B['customer delivery'].\
                loc[subfleets1B['Blade']=='2929M96P01'].iloc[0]))
    if thrust_f =='LEAP-1C30':
        return datetime.datetime.fromisoformat(str(subfleets1C['customer delivery'].\
                loc[subfleets1C['Blade']=='Redesign'].iloc[0]))

def MHPTS1S_start (thrust_f):
```

```
    if thrust_f =='LEAP-1A32':
        return datetime.datetime.fromisoformat(str(subfleets1A['customer delivery'].\
                loc[subfleets1A['Shroud']=='P04 Neutral China/P05 India ME'].iloc[0]))
    if thrust_f =='LEAP-1B28':
        return datetime.datetime.fromisoformat(str(subfleets1B['customer delivery'].\
                loc[subfleets1B['Shroud']=='2790M11P05'].iloc[0]))
    if thrust_f =='LEAP-1C30':
        return datetime.datetime.fromisoformat(str(subfleets1C['customer delivery'].\
                loc[subfleets1C['Shroud']=='2790M11P05'].iloc[0]))


def HPT_digits(ESN_f,bucket_f,hundreds_f):
    if bucket_f == 'HPT':
        table1 = HPT_by_ESN
        table2 = HPT_by_subfleet
    if bucket_f == 'HPT_S1_Shroud':
        table1 = HPTS_by_ESN
        table2 = HPTS_by_subfleet
    if bucket_f == 'HPT_S1_Blade':
        table1 = HPTB_by_ESN
        table2 = HPTB_by_subfleet
    if bucket_f == 'Other':
        table1 = Other_by_ESN
        table2 = Other_by_subfleet
    if bucket_f == 'CDN':
        table1 = CDN_by_ESN
        table2 = CDN_by_subfleet



    progression = table1['progression'].loc[table1['ESN'] == ESN_f].iloc[0]
    x = table2.loc[(table2['subfleet'] == hundreds_f) & (table2['progression'] == progression)]


    if x['tens_new'].iloc[0] != 'copy':
        return hundreds_f + x['tens_new'].iloc[0] * 10 + 1
    if x['tens_new'].iloc[0] == 'copy':
        return hundreds_f + table2['tens_new'].loc[(table2['tens']==x['parent prog'].iloc[0]) &\
                (table2['subfleet']==hundreds_f)].iloc[0] * 10 + x['skip'].iloc[0] + 1
```

```
#########################################################################################################
def subfleet_list_extension (bucket_f):
    subfleet_list = Workscope_Status['ASVN'].loc[Workscope_Status['Failure Distribution Name'] == bucket_f].unique().tolist()
    copy = subfleet_list.copy()
    for fleet in subfleet_list:
        if fleet % 100 < 9:
            for i in range (fleet % 100 + 1,10): copy.append((fleet // 10)*10+i)
    return sorted(list(set(copy)))

#########################################################################################################
def Reliability(x,cur_x,beta,eta,sev):
    try:
        return np.exp(-((x+cur_x)/(eta*sev))**beta)
    except: print(x,cur_x,beta,eta,sev)

def Removal_in_days (sev1, sev2, Eta1, Beta1, Cur_x1, Eta2, Beta2 ,Cur_x2, Eta3, Beta3,\
                     Cur_x3, Eta4, Beta4, Cur_x4, Eta5, Beta5, Cur_x5  ):
    x=1
    while Reliability(x,Cur_x1,Beta1,Eta1,sev1) * Reliability(x,Cur_x2,Beta2,Eta2,sev1) *\
        Reliability(x,Cur_x4,Beta4,Eta4,sev1) * Reliability(x,Cur_x3,Beta3,Eta3,sev2) *\
        Reliability(x,Cur_x5,Beta5,Eta5,sev2)> 0.5: x+=1
    return (x*365.25)/(HRs_ut/FL)
#########################################################################################################
def factor_QT_Blades (thrust_f, region_f, configuration_f, subfleets_f, parameter_f):

    if region_f == 'Middle East': k = 0.85
    else: k = 0.75

    if configuration_f == 'Blade_Redesign_Group':
        if parameter_f == 'Eta': return 999999
        if parameter_f == 'Beta': return 50
    if configuration_f == 'Block': subfleet = 100
    if configuration_f == 'Turkey': subfleet = 200

    if subfleets_f%10 > 1 and parameter_f== 'Eta':
```

```
        return k * qW['Eta'].loc[(qW['engine']==thrust_f) & (qW['Region']==region_f)&\
                (qW['subfleet']==subfleet) & (qW['QT']==1) &\
                (qW['WEibull']=='HPT_S1_Blade')].iloc[0]
    else:
        return qW[parameter_f].loc[(qW['engine']==thrust_f) & (qW['Region']==region_f)&\
                (qW['subfleet']==subfleet) & (qW['QT']==1) &\
                (qW['WEibull']=='HPT_S1_Blade')].iloc[0]
##########################################################################################
def configuration_HPTB_check (thrust_f, date_f):

    if thrust_f == 'LEAP-1B28' and date_f > datetime.datetime.fromisoformat(str(subfleets1B['customer delivery'].\
            loc[subfleets1B['ICAM ID']==800].iloc[0])).date() :return 'Blade_Redesign_Group'
    if thrust_f == 'LEAP-1A32' and date_f > datetime.datetime.fromisoformat(str(subfleets1A['customer delivery'].\
            loc[subfleets1A['ICAM ID']==700].iloc[0])).date() :return 'Blade_Redesign_Group'
    if thrust_f == 'LEAP-1C30' and date_f > datetime.datetime.fromisoformat(str(subfleets1C['customer delivery'].\
            loc[subfleets1C['ICAM ID']==800].iloc[0])).date() :return 'Blade_Redesign_Group'
    else: return 'Turkey'
##########################################################################################
def factor_HPT (thrust_f1,thrust_f2, subfleet_f, region_f, parameter_f, date_f):

    if region_f == 'Middle East': region_f = 'Middle_East'
    if region_f == 'Neutral': factor_r = 0.75 # rest 0.75
    if region_f != 'Neutral': factor_r = 0.9 # middle east 0.85

    if thrust_f1 == 'LEAP-1A32':
        subfleet = subfleets1A
        HPTB_redesign = 700
    if thrust_f1 == 'LEAP-1B28':
        subfleet = subfleets1B
        HPTB_redesign = 800
    if thrust_f1 == 'LEAP-1C30':
        subfleet = subfleets1C
        HPTB_redesign = 800

    if date_f >= datetime.datetime.fromisoformat(str(subfleet['customer delivery'].\
            loc[subfleet['ICAM ID']==900].iloc[0])).date(): factor_f = 1
    if date_f < datetime.datetime.fromisoformat(str(subfleet['customer delivery'].\
            loc[subfleet['ICAM ID']==900].iloc[0])).date(): factor_f = 0.7
    if parameter_f == 'Beta': factor = 1
```

```
    if parameter_f == 'Eta': factor = factor_r * factor_f

    if subfleet_f % 10 == 1:
            return wW[parameter_f].loc[(wW['Engine'] == thrust_f2) &\
                     (wW['Region'] == region_f) &\
                     (wW['Faliure Distribution Name'] == 'HPT') &\
                     (wW['Trigger Value'].str.contains(str(subfleet_f),na=False))].iloc[0]

    if subfleet_f % 10 != 1:
        if date_f < datetime.datetime.fromisoformat(str(subfleet['customer delivery'].\
                loc[subfleet['ICAM ID']==HPTB_redesign].iloc[0])).date():
            return wW[parameter_f].loc[(wW['Engine'] == thrust_f2) &\
                     (wW['Region'] == region_f) &\
                     (wW['Faliure Distribution Name'] == 'HPT') &\
                     (wW['Trigger Value'].str.contains(str((subfleet_f//100)*100+2),na=False))].iloc[0]

        if date_f >= datetime.datetime.fromisoformat(str(subfleet['customer delivery'].\
                loc[subfleet['ICAM ID']==HPTB_redesign].iloc[0])).date():
            return factor * wW[parameter_f].loc[(wW['Engine'] == thrust_f2) &\
                (wW['Region'] == region_f) &\
                (wW['Faliure Distribution Name'] == 'HPT') &\
                (wW['Trigger Value'].str.contains(str(901),na=False))].iloc[0]
##########################################################################################
def configuration_HPTS_check (thrust_f, date_f, region_f):

    if thrust_f == 'LEAP-1B28'  and date_f > datetime.datetime.fromisoformat(str(subfleets1B['customer delivery'].\
            loc[subfleets1B['ICAM ID']==900].iloc[0])).date() :return 'Phase 5'
    if thrust_f == 'LEAP-1A32' and region_f in ['India','Middle East'] and date_f >\
        datetime.datetime.fromisoformat(str(subfleets1A['customer delivery'].loc[subfleets1A['ICAM ID']==800].iloc[0])).date() :\
        return 'Phase 5'
    else: return 'Phase 4'
```

---

```
##########################################################################################################################

def factor_QT_Shrouds (thrust_f, region_f, config_f, Qts_f, parameter_f):
    if 'LEAP-1A' in thrust_f:
        if parameter_f == 'Beta': parameter = 'beta'
        if parameter_f == 'Eta': parameter = 'Eta'
        name = 'name'
```

```
        sW = sW1A
    if 'LEAP-1B' in thrust_f:
        if parameter_f == 'Beta': parameter = 'beta.1'
        if parameter_f == 'Eta': parameter = 'Eta.1'
        name = 'name.1'
        sW = sW1B
    if config_f in ['Phase 2','Phase 3','Phase 4','Phase 5']:
            if Qts_f == 1: return sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+'HPT Stage 1 Shroud'+\
                    ' '+ config_f +' '+'QT1'].iloc[0]
            else:
                if parameter_f == 'Beta': return sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+\
                        'HPT Stage 1 Shroud'+' '+ config_f +' '+'QT1'].iloc[0]
                if parameter_f == 'Eta' :
                    if sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+'HPT Stage 1 Shroud'+\
                            ' '+ config_f +' '+'QT1'].iloc[0] == 999999:
                        return sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+'HPT Stage 1 Shroud'+\
                                ' '+ config_f +' '+'QT1'].iloc[0]
                    else:
                        return 0.75*(sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+'HPT Stage 1 Shroud'+\
                                ' '+ config_f +' '+'QT1'].iloc[0])
    else:
            return sW[parameter].loc[sW[name] == thrust_f+' '+region_f+' '+'HPT Stage 1 Shroud'+\
                    ' '+ config_f +' '+'QT'+str(Qts_f)].iloc[0]

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

def list_reduction(prog_list):

    while (len(prog_list)>1) and (prog_list[-1] == prog_list [-2]):
        prog_list.pop()
    return(prog_list)

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

def HPT_summary(bucket):
    summary_list=[]
    for i in sorted(list(set(HPT_progression['subfleet']))):
        for ESN in list(set(list(HPT_progression['ESN'].loc[HPT_progression['subfleet']==i]))):
            summary_list.append([ESN,i,
```

```
                str(list_reduction(list(HPT_progression['Eta'].loc[(HPT_progression['ESN']== ESN) &\
                                    (HPT_progression['bucket']== bucket) &\
                                    (HPT_progression['subfleet']== i)]))).strip("[]"),
                len(list_reduction(list(HPT_progression['Eta'].loc[(HPT_progression['ESN']== ESN) &\
                                    (HPT_progression['bucket']== bucket) &\
                                    (HPT_progression['subfleet']== i)])))])
summary_by_ESN = pd.DataFrame(summary_list, columns =['ESN','subfleet','progression','lenght'])
summary_by_subfleet = summary_by_ESN.groupby(['subfleet','progression','lenght']).size().reset_index().\
rename(columns={0:'count'}).sort_values(by = ['subfleet', 'lenght'], ascending = [True, True]).reset_index(drop=True)

rows1=[0] #column for preliminary tens
tens = 0
for i in range(1,len(summary_by_subfleet)):
    if summary_by_subfleet['subfleet'][i]==summary_by_subfleet['subfleet'][i-1]: tens = tens+1
    else: tens = 0
    rows1.append(tens)
summary_by_subfleet['tens']=rows1


rows2=[] # column for parent tens indication
rows3=[] # column for skip calculation
for i in range(len(summary_by_subfleet)):
    row1 = 'na'
    row2 = 'na'
    for j in range(len(summary_by_subfleet)-i-1):

        if (summary_by_subfleet['subfleet'][i] == summary_by_subfleet['subfleet'][j]) and\
                (summary_by_subfleet['progression'][i] in summary_by_subfleet['progression'][j]) and\
                (summary_by_subfleet['lenght'][j]-summary_by_subfleet['lenght'][i]>0):
            row1 = summary_by_subfleet['tens'][j]
            row2 = summary_by_subfleet['lenght'][j]-summary_by_subfleet['lenght'][i]
            continue
    rows2.append(row1)
    rows3.append(row2)
summary_by_subfleet['parent prog'] = rows2
summary_by_subfleet['skip'] = rows3


rows4=[] #column for final tens (dozens)
```

```
    tens_new = 0
    for i in range(len(summary_by_subfleet)):
        if i==0:
            if summary_by_subfleet['parent prog'][i]=='na':
                rows4.append(tens_new)
                tens_new = tens_new+1
            else: rows4.append('copy')
        else:
            if (summary_by_subfleet['subfleet'][i] == summary_by_subfleet['subfleet'][i-1]) and\
                    (summary_by_subfleet['parent prog'][i]=='na'):
                rows4.append(tens_new)
                tens_new = tens_new+1
            if (summary_by_subfleet['subfleet'][i] == summary_by_subfleet['subfleet'][i-1]) and\
                    (summary_by_subfleet['parent prog'][i]!='na'):
                rows4.append('copy')
            if (summary_by_subfleet['subfleet'][i] != summary_by_subfleet['subfleet'][i-1]) and\
                    (summary_by_subfleet['parent prog'][i]=='na'):
                tens_new = 0
                rows4.append(tens_new)
                tens_new = tens_new+1
            if (summary_by_subfleet['subfleet'][i] != summary_by_subfleet['subfleet'][i-1]) and\
                    (summary_by_subfleet['parent prog'][i]!='na'):
                rows4.append('copy')
                tens_new = 0
    summary_by_subfleet['tens_new']=rows4

    return summary_by_ESN, summary_by_subfleet
##########################################################################################################################

def HPT_progressions (file1, file2, bucket):

    rows =[]
    for subfleet in sorted(list(set(Workscope_Status['ASVN'].loc[(Workscope_Status['Failure Distribution Name'] == bucket)]))):
        if subfleet % 10 == 1:
            chain = file2['progression'].loc[(file2['subfleet']==(subfleet//100)*100) &\
                        (file2['tens_new']==(subfleet%100)//10)].iloc[0]
            example = file1['ESN'].loc[(file1['progression'] == chain) & (file1['subfleet'] == (subfleet//100)*100)].iloc[0]
            for i in range(9):
                Eta = HPT_progression['Eta'].loc[(HPT_progression['ESN']== example) &\
```

```
                                    (HPT_progression['bucket']== bucket) & (HPT_progression['iter']== i)].iloc[0]
            Beta = HPT_progression['Beta'].loc[(HPT_progression['ESN']== example) &\
                                    (HPT_progression['bucket']== bucket) & (HPT_progression['iter']== i)].iloc[0]
            Gamma = 0
            row = ['ASVN',subfleet + i, bucket, 'Cycles', 'Weibull', Eta, Beta, Gamma]
            rows.append(row)
    return rows

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


#------------------------------frames preparation-----------------------------------------------
time_before = time.time()

#FM subfleets vs Initial Engine delivery date

ESN_list = pd.read_excel(fsa, sheet_name = 'Engine')['Engine Serial Number'].astype(str).tolist()
Wb_buckets = ['Bearing','CDN','EGT','FAN','FOD','HPC','HPT','HPT_S1_Blade',\
              'HPT_S1_Shroud','High_Cost','LPT','LPTN','Other','TOW_CAP-FL']#list of FM Weibulls


subfleets1A = pd.read_excel(model, sheet_name = 'ENTRY INTO SERVICE',header=19).loc[8:16,'ICAM ID': 'Blade']
subfleets1B = pd.read_excel(model, sheet_name = 'ENTRY INTO SERVICE',header=19).loc[:7,'ICAM ID': 'Blade']
subfleets1C = pd.read_excel(model, sheet_name = 'ENTRY INTO SERVICE',header=19).loc[18:20,'ICAM ID': 'Blade']

#HPTB_prog = pd.read_excel(model, sheet_name = 'HPTS1B GE-SAE Comparison', header=1).loc[:, 'Engine Model':'SV8']
#HPTS_prog = pd.read_excel(model, sheet_name = 'HPTS1S GE-SAE Comparison', header=1).loc[:, 'Engine Model':'SV7']


dW = pd.read_excel(model, sheet_name = 'DURABILITY WEIBULLS', header=3).loc[:1078,'Engine Serie':'Subfleets']   #durability Weibulls list
dW['Subfleets']=dW['Subfleets'].astype(str)
dW['SV#']=dW['SV#'].astype(str)

eW = pd.read_excel(model, sheet_name = 'EGT WEIBULLS ', header=3).loc[:199,'Engine Serie':'Subfleet']    #EGT Weibulls list
eW['Subfleet']=eW['Subfleet'].astype(str)

qW = pd.read_excel(model, sheet_name = 'QT Weibulls and Subfleets', header=0).loc[:5847,'engine':'Threshold2']                    #HP
sW1A = pd.read_excel(model, sheet_name = 'lookup', header=1).loc[:661,'name':'Phase']
sW1B = pd.read_excel(model, sheet_name = 'lookup', header=1).loc[:466,'name.1':'Phase.1']
```

```
cW = pd.read_excel(model, sheet_name = 'Cap weibulls', header=0).loc[:,'engine_model':'gamma']

wW = pd.read_excel(model, sheet_name = 'Weibulls', header=0).loc[:,'Trigger Value':'SF']
wW['Trigger Value']=wW['Trigger Value'].astype(str)

cdnW = pd.read_excel(model, sheet_name = 'CDN', header=0).loc[:,'Engine Serie':'Date']
cdnW[['SV#','Subfleets']]=cdnW[['SV#','Subfleets']].astype(str)
#for chceck if CDN kit upgrade were done
CDN_kit_upgrade = pd.DataFrame(ESN_list, columns=['ESN'])
CDN_kit_upgrade['flag']=False

print("Data frames prepare in {} seconds. Total time {} seconds\n".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
#-------------------------Aircraft-----------------------------------------------------------
Aircraft = pd.read_excel(fsa, sheet_name = 'Aircraft')
#---------------------------2.Engine-------------------------------------------------------------

Engine = pd.read_excel(fsa, sheet_name = 'Engine')
Engine['Engine Serial Number']=Engine['Engine Serial Number'].astype(str)
#Engine['Delivery Date']=Engine['Delivery Date'].astype(str)

#SVC data
if '.csv' in svc:
    data = pd.read_csv(svc, delimiter=',')

    #data['ESN'] = pd.to_str(data['ESN'])
#    for index in data.index:
#        data.at[index,'ESN']=data['ESN'].str.replace('"','').str.replace('=','').str.replace(' ','')[index]
if '.xlsx' in svc:
    print('Recognized format of SVC file is \".xlsx\". Please write name of the sheet with data.')
    sheet = input()

    data = pd.read_excel(svc, sheet_name = sheet)
    data['ESN'] = data['ESN'].astype(str).tolist()
    data['Initial Engine Delivery Date'] = data['Initial Engine Delivery Date'].astype(str)
data['Event Date'] =  pd.to_datetime(data['Event Date'], format='%m/%d/%Y')


#region
```

```
for ESN in ESN_list:
    try:
        region = data['Operating region'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0]
#         if region!=region: continue
        if region in ['Neutral','China','India','Middle East']: break
    except: continue

print('Recognized region: ', region, ' - Are you ok? (y/n)')
confirmation_1 = input()
while confirmation_1 != 'y' and confirmation_1 != 'n':
    confirmation_1 = input('Please use correct answear - (y/n)\n')
if confirmation_1 == 'n':
    confirmation_2 = 'n'
    while confirmation_2 not in ['Neutral','China','India','Middle East']:
        confirmation_2 = input('Choose and write region (Neutral/China/India/Middle East)\n')
    region = confirmation_2


#subfleets1B.at[6,'ICAM ID']=800
#if region in['Neutral','China']: subfleets1B.at[6,'ICAM ID']=700
#if region in ['India','Middle East']: subfleets1B.at[6,'ICAM ID']=800

#thrust
for ESN in ESN_list:
    try:
        if data['Engine Model'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0] == 'LEAP-1A':
            thrust1 = 'LEAP-1A32'
            thrust2 = 'LEAP-' + data['Engine Series'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0].replace("'","")
            thrust3 = 'LEAP-1A'
            break
        if data['Engine Model'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0] == 'LEAP-1B':
            thrust1 = 'LEAP-1B28'
            thrust2 = 'LEAP-' + data['Engine Series'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0].replace("'","")
            thrust3 = 'LEAP-1B'
            break
        if data['Engine Model'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0] == 'LEAP-1C':
            thrust1 = 'LEAP-1C30'
            thrust2 = 'LEAP-' + data['Engine Series'].loc[(data['ESN'] == ESN) & (data['Event Type'] == 'CES')].iloc[0].replace("'","")
            thrust3 = 'LEAP-1C'
```

```
            break
    except: continue

possible_thrusts = sorted(list(set(eW['Engine Serie'])))
print('Recognized thrust: ', thrust2, ' - Are you ok? (y/n)')
confirmation_3 = input()
while confirmation_3 != 'y' and confirmation_3 != 'n':
    confirmation_3 = input('Please use correct answear - (y/n)\n')
if confirmation_3 == 'n':
    confirmation_4 = 'n'
    while confirmation_4 not in possible_thrusts:
        print('Choose and write thrust from following list: ',possible_thrusts,'\n')
        confirmation_4 = input()
    thrust2 = confirmation_4
    thrust3 = thrust2[:7]
    if thrust2[:7] == 'LEAP-1A': thrust1 = 'LEAP-1A32'
    if thrust2[:7] == 'LEAP-1B': thrust1 = 'LEAP-1B28'
    if thrust2[:7] == 'LEAP-1C': thrust1 = 'LEAP-1C30'

#CMR cut-off date
ESN_status = pd.read_excel(fsa, sheet_name = 'Engine')['Status Date']
for index in ESN_status.index: ESN_status.at[index]=datetime.datetime.strptime(ESN_status[index],"%m/%d/%Y")
CMR_start = ESN_status.min()


#Current CSV replacement (events number)
cmr_cup = data.loc[(data['Event Type'].isin(['QT','SV','Module SV'])) &\
    (data['Event Date']<CMR_start) & (data['ESN'].isin(ESN_list))]
cmr_cup_SV = data.loc[(data['Event Type'] =='SV') & (data['Event Date']<CMR_start) & (data['ESN'].isin(ESN_list))]

cmr_ces = data.loc[(data['Event Type'].isin(['CES'])) & (data['ESN'].isin(ESN_list)) &\
    data['Conf_in.HPT Stage 1 Blade Group'].notnull() & data['Conf_in.HPT Stage 1 Shroud Group'].notnull()]

start_conf = data.loc[(data['Event Type'].isin(['QT','SV','Module SV','CES']) & (data['ESN'].isin(ESN_list)))]


#two version each of them can be not managed in some Python version
#-------------------- CSN / ASVN / RSV ------------------------------------------------------------
try:
```

```
    event_no = cmr_cup.groupby(['ESN']).size().reset_index().rename(columns={0:'count'})
    for ESN in event_no['ESN']:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        Engine.at[index,'Current CSV#'] = event_no['count'].loc[event_no['ESN']==ESN].iloc[0]

    event_no_SV = cmr_cup_SV.groupby(['ESN']).size().reset_index().rename(columns={0:'count'})
    for ESN in ESN_list:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        try:
            Engine.at[index,'Current RSV#'] = event_no_SV['count'].loc[event_no_SV['ESN']==ESN].iloc[0]
        except:
            Engine.at[index,'Current RSV#'] = 0
    #ASVN replacement (events number + hundreds)
    for ESN in Engine['Engine Serial Number']:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        Engine.at[index,'Current ASV#'] = Engine['Current CSV#'][index] + hundreds(ESN,thrust1)
except:
    pd.options.mode.chained_assignment = None

    event_no = cmr_cup.groupby(['ESN']).size().reset_index().rename(columns={0:'count'})
    for ESN in event_no['ESN']:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        Engine['Current CSV#'][index] = event_no['count'].loc[event_no['ESN']==ESN].iloc[0]
    event_no_SV = cmr_cup_SV.groupby(['ESN']).size().reset_index().rename(columns={0:'count'})
    for ESN in ESN_list:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        try:
            Engine['Current RSV#'][index] = event_no_SV['count'].loc[event_no_SV['ESN']==ESN].iloc[0]
        except:
            Engine['Current RSV#'][index] = 0
    #ASVN replacement (events number + hundreds)
    for ESN in Engine['Engine Serial Number']:
        index = Engine.loc[Engine['Engine Serial Number']==ESN].index
        Engine['Current ASV#'][index] = Engine['Current CSV#'][index] + hundreds(ESN,thrust1)
 #-------------------------------------------------------------------------------------------------

#--------------------------3.LLP-----------------------------------------------------------------

time_before = time.time()
```

```
LLP_Status = pd.read_excel(fsa, sheet_name = 'LLP Status')

print("\nAircraft/Engine/LLP output prepared in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
#----------------------------6.Hardware configuration-----------------------------------------
time_before = time.time()


ESNs_upgraded = sorted(set(cmr_cup['ESN']))
ESNs_CES_only = sorted(set(cmr_ces['ESN'])-set(ESNs_upgraded))
future_delivery = sorted(set(ESN_list)-(set(ESNs_upgraded) | set(ESNs_CES_only)))


HW = []
if thrust1 == 'LEAP-1A32':

    for ESN in ESNs_upgraded:
        event_out = cmr_cup.loc[cmr_cup['ESN']==ESN].loc[cmr_cup['Event Date']==\
                                max(cmr_cup['Event Date'].loc[cmr_cup['ESN']==ESN])].iloc[0]
        HW.append([ESN,region,thrust2, event_out['Initial Engine Delivery Date'],'based SVC',
                   event_out['Conf_out.HPT Stage 1 Blade'],
                   event_out['Conf_out.HPT Stage 1 Blade Group'],
                   event_out['Conf_out.HPT Stage 1 Shroud'],
                   event_out['Conf_out.HPT Stage 1 Shroud Group'],
                   event_out['Conf_out.Campaign QT - NSV Group'],
                   event_out['Conf_out.Campaign QT - HPT S2Nozzle Group'],
                   event_out['Conf_out.Campaign QT - Chipped Shrouds Group'],
                   event_out['Conf_out.Campaign QT - B-sump Group']])
    for ESN in ESNs_CES_only:
        CES_in = cmr_ces.loc[cmr_ces['ESN']==ESN].iloc[0]
        HW.append([ESN,region,thrust2,cmr_ces['Initial Engine Delivery Date'].loc[cmr_ces['ESN']==ESN].iloc[0],'EDS_SVC data',
                   CES_in['Conf_in.HPT Stage 1 Blade'],
                   CES_in['Conf_in.HPT Stage 1 Blade Group'],
                   CES_in['Conf_in.HPT Stage 1 Shroud'],
                   CES_in['Conf_in.HPT Stage 1 Shroud Group'],
                   CES_in['Conf_in.Campaign QT - NSV Group'],
                   CES_in['Conf_in.Campaign QT - HPT S2Nozzle Group'],
                   CES_in['Conf_in.Campaign QT - Chipped Shrouds Group'],
```

```
                CES_in['Conf_in.Campaign QT - B-sump Group']])
    for ESN in future_delivery:
        HW.append([ESN,region,thrust2,Engine['Delivery Date'].loc[Engine['Engine Serial Number']==ESN].iloc[0],'FM Assumption',
                HPTS1B_conf(hundreds(ESN, thrust1), region, thrust1)[1],
                HPTS1B_conf(hundreds(ESN, thrust1), region, thrust1)[0],
                shroud_conf(hundreds(ESN, thrust1), region, thrust1)[1],
                shroud_conf(hundreds(ESN, thrust1), region, thrust1)[0],
                'not impacted','not impacted','not impacted','not impacted'])

    Hardware_Configuration = pd.DataFrame(HW, columns=['ESN','Region','Engine Series',
                                                       'Initial Engine Delivery Date',
                                                       'Configuration Status',
                                                       'Conf_in.HPT Stage 1 Blade',
                                                       'Conf_in.HPT Stage 1 Blade Group',
                                                       'Conf_in.HPT Stage 1 Shroud',
                                                       'Conf_in.HPT Stage 1 Shroud Group',
                                                       'Conf_in.Campaign QT - NSV Group',
                                                       'Conf_in.Campaign QT - HPT S2Nozzle Group',
                                                       'Conf_in.Campaign QT - Chipped Shrouds Group',
                                                       'Conf_in.Campaign QT - B-sump Group'])
if thrust1 == 'LEAP-1B28':

    for ESN in ESNs_upgraded:
        event_out = cmr_cup.loc[cmr_cup['ESN']==ESN].loc[cmr_cup['Event Date']==\
                                max(cmr_cup['Event Date'].loc[cmr_cup['ESN']==ESN])].iloc[0]
        HW.append([ESN,region,thrust2, event_out['Initial Engine Delivery Date'],'based SVC',
                event_out['Conf_out.HPT Stage 1 Blade'],
                event_out['Conf_out.HPT Stage 1 Blade Group'],
                event_out['Conf_out.HPT Stage 1 Shroud'],
                event_out['Conf_out.HPT Stage 1 Shroud Group']])

    for ESN in ESNs_CES_only:
        CES_in = cmr_ces.loc[cmr_ces['ESN']==ESN].iloc[0]
        HW.append([ESN,region,thrust2,cmr_ces['Initial Engine Delivery Date'].loc[cmr_ces['ESN']==ESN].iloc[0],'EDS_SVC data',
                CES_in['Conf_in.HPT Stage 1 Blade'],
                CES_in['Conf_in.HPT Stage 1 Blade Group'],
                CES_in['Conf_in.HPT Stage 1 Shroud'],
                CES_in['Conf_in.HPT Stage 1 Shroud Group']])
```

```
    for ESN in future_delivery:
        HW.append([ESN,region,thrust2,Engine['Delivery Date'].loc[Engine['Engine Serial Number']==ESN].iloc[0],'FM Assumption',
                   HPTS1B_conf(hundreds(ESN, thrust1), region, thrust1)[1],
                   HPTS1B_conf(hundreds(ESN, thrust1), region, thrust1)[0],
                   shroud_conf(hundreds(ESN, thrust1), region, thrust1)[1],
                   shroud_conf(hundreds(ESN, thrust1), region, thrust1)[0]])

    Hardware_Configuration = pd.DataFrame(HW, columns=['ESN','Region','Engine Series',
                                                       'Initial Engine Delivery Date',
                                                       'Configuration Status',
                                                       'Conf_in.HPT Stage 1 Blade',
                                                       'Conf_in.HPT Stage 1 Blade Group',
                                                       'Conf_in.HPT Stage 1 Shroud',
                                                       'Conf_in.HPT Stage 1 Shroud Group'])


print("HW output prepared in {} seconds, Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
time_before = time.time()


# if thrust1 == 'LEAP-1C30'

#Shroud configuration support frames
HPTS_up_list = cmr_cup.loc[(cmr_cup['Event Type'].astype(str) == 'SV') |\
                           (cmr_cup['Wrkscp.HPT_S1_Shroud'] == 3) |\
                           ((cmr_cup['Event Type'].astype(str) == 'Module SV' ) &\
                            (cmr_cup['FM Weibull Bucket'].astype(str) == 'HPT_S1_Shroud'))].\
                            sort_values(['ESN', 'Event Date'], ascending=[True, True]).reset_index()

HPTS_P2_list=[]
for ESN in ESN_list:
    if Shroud_conf_round (Hardware_Configuration['Conf_in.HPT Stage 1 Shroud Group'].\
                          loc[Hardware_Configuration['ESN'] == ESN].iloc[0]) == 2:
        HPTS_P2_list.append(ESN)

HPT_P0P1_NotModified_list= sorted(set(ESN_list)-set(HPTS_P2_list)-set(HPTS_up_list['ESN']))
HPT_P0P1_Modified_list = sorted(set(HPTS_up_list['ESN'])-set(HPTS_P2_list))
```

```
Shroud_QT_conf_mix = QT_counter_mix ()
HTPS1B_QT_conf_mix = QT_counter_mix2 ()

Shroud_QT_conf_mix[0].to_excel('check_HPTS_not_modified.xlsx', index = False, sheet_name='by_ESN')

print("Shrouds/Blades support frames preapred in {} seconds. Total time {} seconds\n".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
print('File \"check_HPTS_not_modified.xlsx\" is generated in working directory.')
print('In case bug occure after this section, please verify generated file and replace non valid progression by correct one.')
print('Change name of file to: \"check_HPTS.xlsx\" and run tool again.')
print('(If bug not occured you don\'t need to do enything)\n')
######################################### HPT progression ###################################

time_before = time.time()

tabela = []
for ESN in ESN_list:
    time_before = time.time()

    date_wo_TAT = date_TAT = datetime.datetime.strptime(Engine['Delivery Date'].\
            loc[Engine['Engine Serial Number'] == ESN].iloc[0],'%m/%d/%Y').date()
    HPTB_conf = HTPS1B_QT_conf_mix[0]['configuration'].loc[HTPS1B_QT_conf_mix[0]['ESN']==ESN].iloc[0]
    HPTS_conf = Shroud_QT_conf_mix[0]['configuration'].loc[Shroud_QT_conf_mix[0]['ESN']==ESN].iloc[0]
    Cur_x1 = clock(ESN,'HPT','ECSN')
    Cur_x2 = clock(ESN,'HPT_S1_Blade','ECSN')
    Cur_x3 = clock(ESN,'HPT_S1_Shroud','ECSN')
    Cur_x4 = clock(ESN, 'CDN', 'ECSN')
    Cur_x5 = clock(ESN, 'Other', 'ECSN')

    subfleet_HPT = hundreds(ESN, thrust1) + counter(ESN,'HPT')+1
    subfleet_HPTB = hundreds(ESN, thrust1) + counter(ESN,'HPT_S1_Blade')+1
    subfleet_HPTS = hundreds(ESN, thrust1) + counter(ESN,'HPT_S1_Shroud')+1
    subfleet_CDN = hundreds(ESN, thrust1) + counter(ESN,'CDN')+1
    subfleet_other = hundreds(ESN, thrust1) + counter(ESN,'Other')+1

    for i in range (9):

        Eta1 = factor_HPT (thrust1, thrust2, subfleet_HPT +i, region, 'Eta', date_wo_TAT)
```

```
Beta1 = factor_HPT (thrust1, thrust2, subfleet_HPT +i, region, 'Beta', date_wo_TAT)
Eta2 = factor_QT_Blades (thrust2, region, HPTB_conf, subfleet_HPTB +i, 'Eta')
Beta2 = factor_QT_Blades (thrust2, region, HPTB_conf, subfleet_HPTB +i, 'Beta')
Eta4 = factor_cdn (ESN, thrust2, subfleet_CDN +i, region, date_wo_TAT, 'Eta')
Beta4 = factor_cdn (ESN, thrust2, subfleet_CDN +i, region, date_wo_TAT, 'Beta')
Eta5 = factor_other (thrust1, subfleet_other +i, region, date_wo_TAT, 'Scale (cycles)')
Beta5 = factor_other (thrust1, subfleet_other +i, region, date_wo_TAT, 'Shape')

if i==0:
    try:
        Eta3 = factor_QT_Shrouds (thrust2, region, HPTS_conf, counter(ESN,'HPT_S1_Shroud') +i +1, 'Eta')
        Beta3 = factor_QT_Shrouds (thrust2, region, HPTS_conf, counter(ESN,'HPT_S1_Shroud') +i +1, 'Beta')
    except:
        from_file = pd.read_excel('check_HPTS.xlsx', sheet_name = 'by_ESN')
        Eta3 = factor_QT_Shrouds (thrust2, region, from_file['configuration'].\
            loc[from_file['ESN']==ESN].iloc[0], from_file['counter'].loc[from_file['ESN']==ESN].iloc[0] +1, 'Eta')
        Beta3 = factor_QT_Shrouds (thrust2, region, from_file['configuration'].\
            loc[from_file['ESN']==ESN].iloc[0],from_file['counter'].loc[from_file['ESN']==ESN].iloc[0] +1, 'Beta')
else:
    Eta3 = factor_QT_Shrouds (thrust2, region, HPTS_conf, counter(ESN,'HPT_S1_Shroud') +i +1, 'Eta')
    Beta3 = factor_QT_Shrouds (thrust2, region, HPTS_conf, counter(ESN,'HPT_S1_Shroud') +i +1, 'Beta')


wiersz = [ESN,'HPT', hundreds(ESN, thrust1), i, Eta1, Beta1, date_wo_TAT ]
tabela.append(wiersz)
wiersz = [ESN,'HPT_S1_Blade', hundreds(ESN, thrust1), i, Eta2, Beta2, date_wo_TAT]
tabela.append(wiersz)
wiersz = [ESN,'HPT_S1_Shroud', hundreds(ESN, thrust1), i, Eta3, Beta3, date_wo_TAT]
tabela.append(wiersz)
wiersz = [ESN,'CDN', hundreds(ESN, thrust1), i, Eta4, Beta4, date_wo_TAT]
tabela.append(wiersz)
wiersz = [ESN,'Other', hundreds(ESN, thrust1), i, Eta5, Beta5, date_wo_TAT]
tabela.append(wiersz)

date_wo_TAT = date_TAT + datetime.timedelta(days = Removal_in_days (sev1, sev2, Eta1, Beta1,\
    Cur_x1, Eta2, Beta2 ,Cur_x2, Eta3, Beta3, Cur_x3, Eta4, Beta4, Cur_x4, Eta5, Beta5, Cur_x5))
HPTB_conf = configuration_HPTB_check (thrust1, date_wo_TAT)
HPTS_conf = configuration_HPTS_check (thrust1, date_wo_TAT, region)
```

```
        date_TAT = date_wo_TAT + datetime.timedelta(days = TAT)
        Cur_x1 = 0
        Cur_x2 = 0
        Cur_x3 = 0
        Cur_x4 = 0
        Cur_x5 = 0
    print(ESN, "\t progression calculated in {} seconds. Total time {} seconds".\
          format(round(time.time()-time_before,1), round(time.time()-start_time,1)))

HPT_progression = pd.DataFrame(tabela,columns=['ESN','bucket', 'subfleet', 'iter', 'Eta', 'Beta', 'date_wo_TAT'])
print("\nHPT progression prepared in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))

################################### HPT additional frmames ###################################

time_before = time.time()

HPT_by_ESN = HPT_summary('HPT')[0]
HPT_by_subfleet = HPT_summary('HPT')[1]
HPTS_by_ESN = HPT_summary('HPT_S1_Shroud')[0]
HPTS_by_subfleet = HPT_summary('HPT_S1_Shroud')[1]
HPTB_by_ESN = HPT_summary('HPT_S1_Blade')[0]
HPTB_by_subfleet = HPT_summary('HPT_S1_Blade')[1]
CDN_by_ESN = HPT_summary('CDN')[0]
CDN_by_subfleet = HPT_summary('CDN')[1]
Other_by_ESN = HPT_summary('Other')[0]
Other_by_subfleet = HPT_summary('Other')[1]



print("HPTB/HPTS frames preapred in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
#############################################################################################

#----------------------------4.Workscope status-----------------------------------------------
time_before = time.time()

rows= []
for ESN in ESN_list:
```

```
    for bucket in Wb_buckets:
        if bucket in ['HPT','HPT_S1_Shroud','HPT_S1_Blade', 'Other', 'CDN']:
            row = [ESN, bucket, HPT_digits(ESN,bucket,hundreds(ESN, thrust1)),\
                   clock(ESN,bucket,'ECSN'), clock(ESN,bucket,'ETSN')]
            rows.append(row)
        else:
            row = [ESN, bucket, hundreds(ESN, thrust1)+ dozens(ESN, bucket)+\
                   counter(ESN,bucket)+1, clock(ESN,bucket,'ECSN'), clock(ESN,bucket,'ETSN')]
            rows.append(row)
Workscope_Status = pd.DataFrame(rows,columns=['Engine Serial Number',
                                              'Failure Distribution Name',
                                              'ASVN','Cycles Since Visit',
                                              'Time Since Visit (hours)'])


print("Workscope status prepared in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))

#----------------------------5.Workscope Failure-----------------------------------------
time_before = time.time()

rows2= []
for bucket in Wb_buckets:
    row2 = ['ASVN', 1, bucket, 'Cycles', 'Weibull', 999999, 50, 0]
    rows2.append(row2)

for bucket in Wb_buckets:

    if bucket not in ['EGT','TOW_CAP-FL','HPT_S1_Blade','HPT_S1_Shroud','HPT','CDN','Other']:  ########################
#         if bucket in ['CDN','HPT']: thrust = thrust2
        thrust = thrust1
        subfleet_list = subfleet_list_extension(bucket)

        for subfleet in subfleet_list:
            Eta = factor_durability (thrust, subfleet, region, bucket, 'Scale (cycles)')
            Beta = factor_durability (thrust, subfleet, region, bucket, 'Shape')
            Gamma = 0
            row2 = ['ASVN',subfleet, bucket, 'Cycles','Weibull', Eta, Beta, Gamma]
            rows2.append(row2)
```

```
    if bucket == 'EGT':#########################################################################################################
        subfleet_list = subfleet_list_extension(bucket)
        for subfleet in subfleet_list:
            Eta = factor_egt (thrust2, subfleet, region, 'Scale (cycles)')
            Beta = factor_egt (thrust2, subfleet, region, 'Shape')
            Gamma = factor_egt (thrust2, subfleet, region, 'Offset (cycles)')
            row2 = ['ASVN',subfleet, bucket, 'Cycles', 'Weibull', Eta, Beta, Gamma]
            rows2.append(row2)

    if bucket == 'TOW_CAP-FL': ################################################################################################
        subfleet_list = subfleet_list_extension(bucket)
        for subfleet in subfleet_list:
            Eta = factor_cap (thrust2,'eta (hours)', subfleet)
            Beta = factor_cap (thrust2,'beta', subfleet)
            Gamma = factor_cap (thrust2,'gamma', subfleet)
            row2 = ['ASVN',subfleet, bucket, 'Cycles', 'Weibull', Eta/2, Beta, Gamma]
            rows2.append(row2)

    if bucket == 'HPT_S1_Blade':
        rows2.extend(HPT_progressions (HPTB_by_ESN, HPTB_by_subfleet, 'HPT_S1_Blade'))

    if bucket =='HPT_S1_Shroud':
        rows2.extend(HPT_progressions (HPTS_by_ESN, HPTS_by_subfleet, 'HPT_S1_Shroud'))

    if bucket == 'HPT':
        rows2.extend(HPT_progressions (HPT_by_ESN, HPT_by_subfleet, 'HPT'))

    if bucket == 'CDN':
        rows2.extend(HPT_progressions (CDN_by_ESN, CDN_by_subfleet, 'CDN'))

    if bucket == 'Other':
        rows2.extend(HPT_progressions (Other_by_ESN, Other_by_subfleet, 'Other'))



for bucket in Wb_buckets: #for ICAM
    row2 = ['ASVN', 9999, bucket, 'Cycles', 'Weibull', 999999, 50, 0]
```

```
    rows2.append(row2)

Workscope_Failure = pd.DataFrame(rows2,columns=['Trigger Type',
                                                'Trigger Value',
                                                'Failure Distribution Name',
                                                'Unit',
                                                'Distribution Curve Type',
                                                'Eta',
                                                'Beta',
                                                'T-Zero'])


HPTS3 = []
for i in ESN_list:
    if Shroud_QT_conf_mix[0]['configuration'].loc[Shroud_QT_conf_mix[0]['ESN'] == i].iloc[0] == 'P2':
        HPTS3.append([i,'P2'])
    else:
        HPTS3.append([i,thrust2 +' '+ region +' '+ 'HPT Stage 1 Shroud' +' '+\
            Shroud_QT_conf_mix[0]['configuration'].loc[Shroud_QT_conf_mix[0]['ESN'] == i].iloc[0] +' '+\
            'QT'+str((Shroud_QT_conf_mix[0]['counter'].loc[Shroud_QT_conf_mix[0]['ESN'] == i].iloc[0])+1)])
HPTS_3sheet = pd.DataFrame(HPTS3,columns=['ESN','chain'])

print("Workscope failure prepared in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))

#podmianka sublot 2000/3000/4000

if region == 'China': tousent=2000
if region == 'India': tousent=3000
if region == 'Middle East': tousent=4000
if region == 'Neutral': tousent=0

Engine['Current ASV#']=Engine['Current ASV#'] + tousent
Workscope_Status['ASVN']=Workscope_Status['ASVN'] + tousent
Workscope_Failure['Trigger Value']=Workscope_Failure['Trigger Value'] + tousent

#------------------------------Excel file preparation - final step-----------------------------
time_before = time.time()
with pd.ExcelWriter('FSA_update.xlsx') as writer:
    Aircraft.to_excel(writer, index = False, sheet_name='Aircraft')                    #1
```

```
    Engine.to_excel(writer, index = False, sheet_name='Engine')                                    #2
    LLP_Status.to_excel(writer, index = False, sheet_name='LLP Status')                            #3
    Workscope_Status.to_excel(writer, index = False, sheet_name='Workscope Status')                #4
    Workscope_Failure.to_excel(writer, index = False, sheet_name='Workscope Failure')              #5
    Hardware_Configuration.to_excel(writer, index = False, sheet_name='Hardware Configuration') #6
    Shroud_QT_conf_mix[0].to_excel(writer, index = False, sheet_name='HPTS1-1')
    Shroud_QT_conf_mix[1].to_excel(writer, index = False, sheet_name='HPTS1-2')
    HPTS_3sheet.to_excel(writer, index = False, sheet_name='HPTS1-3')
    HTPS1B_QT_conf_mix[0].to_excel(writer, index = False, sheet_name='HPTB1-1')
    HTPS1B_QT_conf_mix[1].to_excel(writer, index = False, sheet_name='HPTB1-2')
    HPT_progression.to_excel(writer, index = False, sheet_name='HPT_progression')
    HPT_by_ESN.to_excel(writer, index = False, sheet_name='HPT_by_ESN')
    HPT_by_subfleet.to_excel(writer, index = False, sheet_name='HPT_by_subfleet')
    HPTS_by_ESN.to_excel(writer, index = False, sheet_name='HPTS_by_ESN')
    HPTS_by_subfleet.to_excel(writer, index = False, sheet_name='HPTS_by_subfleet')
    HPTB_by_ESN.to_excel(writer, index = False, sheet_name='HPTB_by_ESN')
    HPTB_by_subfleet.to_excel(writer, index = False, sheet_name='HPTB_by_subfleet')
    CDN_by_ESN.to_excel(writer, index = False, sheet_name='CDN_by_ESN')
    CDN_by_subfleet.to_excel(writer, index = False, sheet_name='CDN_by_subfleet')
    Other_by_ESN.to_excel(writer, index = False, sheet_name='Other_by_ESN')
    Other_by_subfleet.to_excel(writer, index = False, sheet_name='Other_by_subfleet')


print("DONE - Output file \"FSA_update.xlsx\" created in {} seconds. Total time {} seconds".\
      format(round(time.time()-time_before,1), round(time.time()-start_time,1)))
```