Introduction

This code organizes data from SOM opps extractions, according to this sequesce:

a) Past 12 months: we capture conversion & duration info for 'won' and 'not won' opps with these slices. We also go deep on the won projects, analysing duration of the opportunities and their value frequency.
BA - business type (merged with Active opps - 'BA - business type');
BU - business type (merged with Active opps - 'BU - business type');
Employee - BA - business type (merged with Active opps - 'Employee - BA - business type');
Employee - BU - business type (merged with Active opps - 'Employee - BU - business type');

b) Active opps: we capture conversion, duration and update info of active opps with these slices:
Macro description of active opps value by BAs BA - business type (output BA_Business_Type_overview)
BU - business type (merged in modified SOM as consolidatedBU)
Employee - BA - business type (output Empl_BA_Type_Overview)
Employee - BU - business type (merged in Modified SOM as instance_summary)

c) Modified SOM: original SOM with additional information such as BAs, modified origin, and aggregated data by BU

Henrique (Ago-25-2022)

In [62]:

```
#Functions

# Function to output data frames
def output_path(file_name, source_df, nome_index):
    output_path = Path(r"C:\Users\P12044\Bühler\BNAM Business Intelligence - Shared doc
    source_df.to_csv(output_path, index=True, encoding = 'utf-8', index_label = nome_in
```

In [63]:

```
# Importing the required libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from datetime import datetime, date, timedelta
from pathlib import Path

#from sklearn.compose import make_column_transformer
#from sklearn.preprocessing import StandardScaler
#from sklearn.ensemble import RandomForestClassifier
#from sklearn.model_selection import GridSearchCV
#from sklearn.model_selection import train_test_split
#from sklearn.metrics import confusion_matrix
#from sklearn.metrics import balanced_accuracy_score
#from sklearn.metrics import classification_report
#from sklearn.metrics import mean_squared_error
```

Getting data

In [64]:
```
# Manual input - SOM file and Lead file names

som_file = '20220826_SOM_opps.csv'
```

In [65]:
```
# File paths for input information
som_file_path = Path(r"C:\Users\P12044\Bühler\BNAM Business Intelligence - Shared docum
#leads_file_path = Path(r"C:\Users\P12044\Bühler\BNAM Business Intelligence - Shared do

# Reading data
som_df = pd.read_csv(som_file_path, encoding = 'utf-8', low_memory=False)
print("Opportunities dataframe shape: ", som_df.shape)
```

```
Opportunities dataframe shape:  (7018, 28)
```

In [66]:
```
#display(som_df.iloc[0,:])

#print(date.today())
```

In [67]:
```
#display(som_df.iloc[0:5,5])
```

In [68]:
```
#Setting the report date
som_df['Report Date'] = date.today()


#Setting the value columns as float type (numeric)

#som_df['Exp. Sales Volume'] = som_df['Exp. Sales Volume'].str.replace('.', '')
som_df['Exp. Sales Volume'] = som_df['Exp. Sales Volume'].str.replace(',', '').astype(f

#som_df['Exp. Sales Vol. in CHF'] = som_df['Exp. Sales Vol. in CHF'].str.replace('.', '
som_df['Exp. Sales Vol. in CHF'] = som_df['Exp. Sales Vol. in CHF'].str.replace(',', ''

#float for chance of realization
som_df['Chance of Realization'] = (som_df['Chance of Realization'].str.replace('%', '')

#float for chance for Buhler
som_df['Chance for Bühler'] = (som_df['Chance for Bühler'].str.replace('%', '').astype(
```

In [69]:
```
#display(som_df.iloc[0,:])
print("Opportunities dataframe shape: ", som_df.shape)
```

```
Opportunities dataframe shape:  (7018, 29)
```

In [70]:
```
dia = str(input('Please enter when you extracted the datasets from CRM (Day/Month/Year)
```

```
Please enter when you extracted the datasets from CRM (Day/Month/Year):02/09/2022
```

In [71]:
```
tag = datetime.strptime(dia, "%d/%m/%Y").strftime('%Y-%m-%d')
date = datetime.strptime(dia, "%d/%m/%Y")
```

```
print(tag, date, sep='\n')
```

```
2022-09-02
2022-09-02 00:00:00
```

In [72]:

```
#display(som_df.iloc[0:10,10:20])
```

In [73]:

```
# Adjusting date columns

som_df['Report Date'] = pd.to_datetime(som_df['Report Date'], dayfirst=True)
som_df['Start Date'] = pd.to_datetime(som_df['Start Date'], dayfirst=True)
som_df['PlanDate: Order Rel.'] = pd.to_datetime(som_df['PlanDate: Order Rel.'], dayfirs
som_df['Changed on'] = pd.to_datetime(som_df['Changed on'], dayfirst=True)

#display((som_df['Report Date']).head())
#display((som_df['Start Date']).head())
#display((som_df['PlanDate: Order Rel.']).head())
```

In [74]:

```
#display(som_df.iloc[1,:])
```

In [75]:

```
#Setting new column with Order Released Month and Opp Started Month

som_df['Month_Order_Released'] = som_df['PlanDate: Order Rel.'].dt.to_period('M')
som_df['Month_Opp_Started'] = som_df['Start Date'].dt.to_period('M')
som_df['Month_Opp_was_Updated'] = som_df['Changed on'].dt.to_period('M')
```

In [76]:

```
# Calculating how many days since the last update

som_df['Days since last update'] = (som_df['Report Date'] - som_df['Changed on']).dt.da
#som_df['Days since last update'] = som_df['Days since last update'].astype(str)
#som_df['Days since last update'] = int(som_df['Days since last update'].strftime("d%")
```

In [77]:

```
# Creating BAs collumn
dataset = som_df

conditions = [(dataset['Business unit']=='AG') | (dataset['Business unit']=='DR'),
              (dataset['Business unit']=='FU') | (dataset['Business unit']=='AN') | (da
              (dataset['Business unit']=='HN') | (dataset['Business unit']=='ES'),

              (dataset['Business unit']=='BA') | (dataset['Business unit']=='MX'),
              (dataset['Business unit']=='MO') | (dataset['Business unit']=='SC'),

              (dataset['Business unit']=='BI') | (dataset['Business unit']=='WF') | (da

              (dataset['Business unit']=='BD') ,
              (dataset['Business unit']=='LO') | (dataset['Business unit']=='LA') |(dat
              (dataset['Business unit']=='GD') | (dataset['Business unit']=='BS') ,

              (dataset['Business unit']=='GS') | (dataset['Business unit']=='GH') | (da

              (dataset['Business unit']=='DA') | (dataset['Business unit']=='DS') | (da
              (dataset['Business unit']=='MU') | (dataset['Business unit']=='SM')
              ]

choices = ['VN', 'VN', 'VN', 'CF', 'CF', 'HAAS', 'DC','LO','GD', 'GQ', 'DT', 'MS']

dataset['BA w HAAS'] = np.select(conditions, choices)

#Checking on the Updated Type data quality
for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1] == p)))
```

```
Count of CF is  1299
Count of DC is  314
Count of DT is  540
Count of GD is  172
Count of GQ is  660
Count of HAAS is  532
Count of LO is  337
Count of MS is  1865
Count of VN is  1299
```

In [78]:

```
#display(dataset.iloc[0,:])
```

In [79]:

```
# Creating BA column

conditions = [(dataset['BA w HAAS'] == 'HAAS'), (dataset['BA w HAAS']!='HAAS')]

choices = ['CF', dataset['BA w HAAS']]

dataset['BA'] = np.select(conditions, choices)

for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1] == p)))
```

```
Count of CF is  1831
Count of DC is  314
Count of DT is  540
Count of GD is  172
Count of GQ is  660
Count of LO is  337
Count of MS is  1865
Count of VN is  1299
```

In [80]:

```
#display(dataset.iloc[0,:])
```

In [81]:

```
# Final Status Thirty

conditions = [(dataset['Status']=='User Error') | (dataset['Status']=='User Error/Dupli
               (dataset['Status']=='Cancelled by Buhler') | (dataset['Status']=='Cancell
               (dataset['Status']=='Active'),
               (dataset['Status']=='Won')
              ]

choices = ['Lost/Cancelled/Other', "Active", 'Won']

dataset['Final_Status'] = np.select(conditions, choices)
```

In [82]:
```
#Checking on the Updated Status data quality

for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1] == p)))
```

```
Count of Active is  2968
Count of Lost/Cancelled/Other is  3055
Count of Won is  995
```

In [83]:
```
#Creating the 'adjusted origin' column
conditions = [(dataset['Origin']=='New Buhler Website'),
              (dataset['Origin']!='New Buhler Website')
                          ]

choices = ['Website', dataset['Origin']]

dataset['adjusted_origin'] = np.select(conditions, choices)

dataset['adjusted_origin'] = dataset['adjusted_origin'].fillna('Undefined')

#print(np.unique(dataset['adjusted_origin']))
```

In [84]:
```
#display(dataset.iloc[1,-5])
```

In [85]:
```
# Splitting Employee and Location
dataset = dataset.rename(columns = {'Employee Responsible':'EmpLOC'})

dataset [['Employee Responsible', 'Loc 1', 'Loc 2']]= dataset.EmpLOC.str.split("/",expa
```

In [86]:
```
# Creating new column: Employee - BA
dataset ['Employee-BA'] = dataset ['Employee Responsible'].astype(str) + '-' + dataset


# Creating new column: Employee - BA - Type
dataset ['Employee-BA-Type'] = dataset ['Employee Responsible'].astype(str) + '-' + dat


# Creating new column: Employee - BU
dataset ['Employee-BU'] = dataset ['Employee Responsible'].astype(str) + '-' + dataset


# Creating new column: Employee - BA - Type
dataset ['Employee-BU-Type'] = dataset ['Employee Responsible'].astype(str) + '-' + dat
```

In [87]:
```
# Creating new column: BA - Business Type
dataset ['BA - Business Type'] = dataset ['BA'].astype(str) + '-' + dataset ['Business


# Creating new column: BU - Business Type
dataset ['BU - Business Type'] = dataset ['Business unit'].astype(str) + '-' + dataset
```

In [88]:
```
# Creating new column: Opp ID - Opp Description
```

```
dataset ['Opp ID - Opp Description'] = dataset ['Opportunity ID'].astype(str) + ' - ' +
```

In [89]:

```
if dataset.columns[13] == "Chance for B�hler":
    dataset.rename(columns = {"Chance for B�hler" : 'Chance for Bühler' }, inplace = Tr

print(dataset.columns[13])
print (dataset.columns[13] == "Chance for B�hler")
```

```
Chance for Bühler
False
```

In [90]:

```
#Removing Business Type Nas
dataset['Business Type']=dataset['Business Type'].fillna('undefined')


#Creating supporting colum: BA_Start_Month
dataset['BA_Start_Month'] = dataset['BA'] + '-' + dataset['Month_Opp_Started'].dt.strft
```

In [91]:

```
#Storing modifications on som_df_modified dataframe
som_df_modified=dataset
```

In [92]:

```
#display(som_df_modified.iloc[0,:])
```

Analyzing past 12 months

In [93]:

```
#Filtering only opps from past 12 months

dataset = som_df_modified[
                            (som_df_modified['PlanDate: Order Rel.'] < date)&
                            (som_df_modified['Final_Status'] != 'Active') #Active opps
                        ]



#saving dataset
file_name ='%s_conversions_last_12_months.csv'%tag
source_df = dataset
nome_index = "#"
output_path(file_name, source_df, nome_index)
```

Aggregating the count and value of opps by BAs & Business Type

In [94]:

```
#Agregating the count and value of opps by BAs & Business Type
chaves = np.unique(dataset['BA - Business Type'])
colunas = list(np.unique(dataset['Final_Status']))

instances_count = {}
instances_value = {}
instances_duration = {}

for a in chaves:
    instances_count[a] = []
    instances_value[a] = []
    instances_duration[a]=[]

for i in chaves:
    BA_cluster = dataset[(dataset['BA - Business Type'] == i)]
    for status in np.unique(BA_cluster['Final_Status']):
        BA_status_cluster = BA_cluster[BA_cluster['Final_Status'] == status ]
        soma_count= BA_status_cluster.shape[0]
        soma_value= round((sum(BA_status_cluster['Exp. Sales Vol. in CHF']))/1000000,2)

        median_duration = BA_status_cluster['Opportunity Duration (No of Days)'].median
        Q95_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'].qu
        max_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'].ma

        instances_count[i].append(soma_count)
        instances_value[i].append(soma_value)
        instances_duration[i].append(median_duration)
        instances_duration[i].append(Q95_duration)
        instances_duration[i].append(max_duration)


#getting the duration median, Q3 and max value of BNAM - Business Type
for a in np.unique(dataset['Business Type']):
    instances_count['BNAM-%s'%a]=[]
    instances_value['BNAM-%s'%a]=[]
    instances_duration['BNAM-%s'%a]=[]

for a in np.unique(dataset['Business Type']):
    type_cluster = dataset[dataset['Business Type'] == a]
    for status in np.unique(type_cluster ['Final_Status']):
            BA_status_cluster = type_cluster[type_cluster['Final_Status'] == status ]
            soma_count= BA_status_cluster.shape[0]
            soma_value= round((sum(BA_status_cluster['Exp. Sales Vol. in CHF']))/100000
            median_duration = BA_status_cluster['Opportunity Duration (No of Days)'].me
            Q95_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'
            max_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'


            instances_count['BNAM-%s'%a].append(soma_count)
            instances_value['BNAM-%s'%a].append(soma_value)
            instances_duration['BNAM-%s'%a].append(median_duration)
            instances_duration['BNAM-%s'%a].append(Q95_duration)
            instances_duration['BNAM-%s'%a].append(max_duration)

#getting the duration median, Q3 and max value for BNAM

instances_count['BNAM-All']=[]
instances_value['BNAM-All']=[]
instances_duration['BNAM-All']=[]

for status in np.unique(type_cluster ['Final_Status']):
        BA_status_cluster = dataset[dataset['Final_Status'] == status ]
        soma_count= BA_status_cluster.shape[0]
        soma_value= round((sum(BA_status_cluster['Exp. Sales Vol. in CHF']))/1000000,2)
        median_duration = BA_status_cluster['Opportunity Duration (No of Days)'].median
        Q95_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'].qu
        max_duration = round((BA_status_cluster['Opportunity Duration (No of Days)'].ma
        instances_count['BNAM-All'].append(soma_count)
        instances_value['BNAM-All'].append(soma_value)
        instances_duration['BNAM-All'].append(median_duration)
        instances_duration['BNAM-All'].append(Q95_duration)
        instances_duration['BNAM-All'].append(max_duration)

## ---------- creating opps. value dataset -----------------------
instances_count=pd.DataFrame.from_dict(instances_count, orient='index', columns=['Count
#display(instances_count.head())
#print()

## ---------- creating opps. value dataset -----------------------

instances_value=pd.DataFrame.from_dict(instances_value, orient='index', columns=['CHF_n

opps_BA_Type_1year = pd.merge(instances_count, instances_value, left_index=True, right_

# adding a total line
#total = opps_1year.sum(numeric_only=True)
#total.name = 'BNAM'
#opps_1year = opps_1year.append(total.transpose())

# calculating the conversion won/(Lost/cancelled/others)
opps_BA_Type_1year['Conversion by $$']=opps_BA_Type_1year['CHF_Won_1y']/opps_BA_Type_1y
opps_BA_Type_1year['Conversion by count']=opps_BA_Type_1year['Count_Won_1y']/opps_BA_Ty

#display(opps_BA_Type_1year.head())

## ---------- creating opps. duration dataset -----------------------
instances_duration=pd.DataFrame.from_dict(instances_duration, orient='index', columns=[
                                                                                  'not_wo
                                                                                  'not_wo
                                                                                  'not_wo
                                                                                  'won_me
                                                                                  'won_95
                                                                                  'won_ma
                                                                                     ])



opps_BA_Type_1year = pd.merge(opps_BA_Type_1year, instances_duration, left_index=True,

#display(opps_BA_Type_1year)
```

In [95]:

```
max_value_conversion = opps_BA_Type_1year['Conversion by $$'].max()
max_count_conversion = opps_BA_Type_1year['Conversion by count'].max()

print('Max value conversion is: ', max_value_conversion)
print('Max count conversion is: ', max_count_conversion)
```

```
Max value conversion is:  3.120689655172414
Max count conversion is:  1.3333333333333333
```

Deep dive on won projects

In [96]:

```
dataset = dataset[dataset['Final_Status'] == "Won"]
```

```
In [97]: #Agregating the value of sold opps by BAs-Business Type and Duration

         chaves = np.unique(dataset['BA - Business Type'])
         #colunas = list(np.unique(dataset['Final_Status']))
         colunas = []

         instances_active = {}

         # getting values for BA - Business Type
         for a in chaves:
             instances_active[a] = []
             #instances_value[a] = []
             #instances_duration[a]=[]


         # getting values for BNAM - Business Type
         for a in np.unique(dataset['Business Type']):
             instances_active['BNAM-%s'%a]=[]
             instances_active['BNAM-%s'%a]=[]
             instances_active['BNAM-%s'%a]=[]

         # getting values for BNAM
         instances_active['BNAM-All']=[]

         coluna = []
         start = 0

         for dias in range (1,16):
             end = (365/2)*dias
             #print(start,end)
             col_label= 'Value (millionCHF) of won opps from %s'%(start/365) + ' to %s'%(end/365
             coluna.append(col_label)
             duration_cluster = dataset[
                                         (dataset['Opportunity Duration (No of Days)'] >= start) &
                                         (dataset['Opportunity Duration (No of Days)'] < end)
                                            ]

             #aggregating by BA - Business Type
             for i in chaves:
                 BA_cluster = duration_cluster[(duration_cluster['BA - Business Type'] == i)]
                 sum_update = round(sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000,2)
                 instances_active[i].append(sum_update)

             #aggregating by Business Type
             for a in np.unique(dataset['Business Type']):
                 BA_cluster = duration_cluster[(duration_cluster['Business Type'] == a)]
                 sum_update = round(sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000,2)
                 instances_active['BNAM-%s'%a].append(sum_update)

             #aggregating by BNAM
             BA_cluster = duration_cluster
             sum_update = round(sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000,2)
             instances_active['BNAM-All'].append(sum_update)

             start=end


         instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

         #display(instances_active)
```

```
#print(instances_active)
#print(coluna)

opps_BA_Type_1year = pd.merge(opps_BA_Type_1year, instances_active, left_index=True, ri
#BAs_consolidated = pd.merge(BAs_consolidated, instances_active, left_index=True, right_
```

In [98]:

```
#Agregating the count of sold opps by BAs-Business Type and Value

chaves = np.unique(dataset['BA - Business Type'])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

# getting values for BA - Business Type
for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]


# getting values for BNAM - Business Type
for a in np.unique(dataset['Business Type']):
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]

# getting values for BNAM
instances_active['BNAM-All']=[]

coluna = []
start = 0
maximo = ((dataset['Exp. Sales Vol. in CHF'].max()+1000000))/1000000

for gaps in range (1,10):

    end = round(maximo/10*gaps,1)
    #print(start,end)
    col_label= 'Count of won opps with values from %s'%(start) + ' to %s'%(end) +' mCHF
    coluna.append(col_label)
    duration_cluster = dataset[
                            (dataset['Exp. Sales Vol. in CHF']/1000000 >= start) &
                            (dataset['Exp. Sales Vol. in CHF']/1000000 < end)
                               ]

    #aggregating by BA - Business Type
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster['BA - Business Type'] == i)]
        sum_update = (BA_cluster['Exp. Sales Vol. in CHF']).count()
        instances_active[i].append(sum_update)

    #aggregating by Business Type
    for a in np.unique(dataset['Business Type']):
        BA_cluster = duration_cluster[(duration_cluster['Business Type'] == a)]
        sum_update = (BA_cluster['Exp. Sales Vol. in CHF']).count()
        instances_active['BNAM-%s'%a].append(sum_update)

    #aggregating by BNAM
    BA_cluster = duration_cluster
    sum_update = (BA_cluster['Exp. Sales Vol. in CHF']).count()
    instances_active['BNAM-All'].append(sum_update)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)

opps_BA_Type_1year = pd.merge(opps_BA_Type_1year, instances_active, left_index=True, ri
```

Aggregating the count and value of opps by BUs & Business Type

In [99]:

```
#Agregating the count and value of opps by BUs & Business Type


dataset = som_df_modified[
                             (som_df_modified['PlanDate: Order Rel.'] < date)&
                             (som_df_modified['Final_Status'] != 'Active') #Active opps
                         ]


chaves = np.unique(dataset['BU - Business Type'])
colunas = list(np.unique(dataset['Final_Status']))

instances_count = {}
instances_value = {}
instances_duration = {}

for a in chaves:
    instances_count[a] = []
    instances_value[a] = []
    instances_duration[a]=[]

for i in chaves:
    BA_cluster = dataset[(dataset['BU - Business Type'] == i)]
    for status in np.unique(BA_cluster['Final_Status']):
        BA_status_cluster = BA_cluster[BA_cluster['Final_Status'] == status ]
        soma_count= BA_status_cluster.shape[0]
        soma_value= (sum(BA_status_cluster['Exp. Sales Vol. in CHF']))/1000000
        median_duration = BA_status_cluster['Opportunity Duration (No of Days)'].median
        Q3_duration = BA_status_cluster['Opportunity Duration (No of Days)'].quantile(0
        max_duration = BA_status_cluster['Opportunity Duration (No of Days)'].max()

        instances_count[i].append(soma_count)
        instances_value[i].append(soma_value)
        instances_duration[i].append(median_duration)
        instances_duration[i].append(Q3_duration)
        instances_duration[i].append(max_duration)


## ---------- creating opps. value dataset -----------------------
instances_count=pd.DataFrame.from_dict(instances_count, orient='index', columns=['Count
#display(instances_count)
#print()
```

```
## ---------- creating opps. value dataset -----------------------

instances_value=pd.DataFrame.from_dict(instances_value, orient='index', columns=['CHF_n

opps_BU_Type_1year = pd.merge(instances_count, instances_value, left_index=True, right_

#display(opps_BA_Type_1year)

# adding a total line
#total = opps_1year.sum(numeric_only=True)
#total.name = 'BNAM'
#opps_1year = opps_1year.append(total.transpose())

# calculating the conversion won/(Lost/cancelled/others)
opps_BU_Type_1year['Conversion by $$']=opps_BU_Type_1year['CHF_Won_1y']/opps_BU_Type_1y
opps_BU_Type_1year['Conversion by Count']=opps_BU_Type_1year['Count_Won_1y']/opps_BU_Ty

#display(opps_BU_Type_1year.head())


## ---------- creating opps. duration dataset -----------------------
instances_duration=pd.DataFrame.from_dict(instances_duration, orient='index', columns=[
                                                                                'not_wo
                                                                                'not_wo
                                                                                'not_wo
                                                                                'won_me
                                                                                'won_75
                                                                                'won_ma
                                                                                   ])

opps_BU_Type_1year = pd.merge(opps_BU_Type_1year, instances_duration, left_index=True, 


#display(opps_BU_Type_1year)
```

Checking 12 months employees' conversion rate by BA and Business Type

In [100…

```
#Aggregating the count and value of opps by Employees
#All business type: CS, P&P and SMB
#All employees
#Aggregated by Employee - Business Area - Business Type

object_matter = 'Employee-BA-Type'
linhas = np.unique(dataset [object_matter])
colunas = list(np.unique(dataset['Final_Status']))
# BA = np.unique(dataset['BA'])

instances_employee_12m = {}
col_name =[]

for a in linhas:
    instances_employee_12m [a] = []

for status in colunas:
    name = "1y_count_" + str(status)
    col_name.append(name)
    name = "1y_sum_$$_" + str(status)
    col_name.append(name)

    for employee in linhas:
        status_cluster = dataset[ (dataset['Final_Status'] == status) & (dataset[object
        soma_count = status_cluster.shape[0]
        soma_value = (sum(status_cluster['Exp. Sales Vol. in CHF']))/1000000
        #median_value = (status_cluster['Exp. Sales Vol. in CHF'].median())/1000000
        instances_employee_12m[employee].append(soma_count)
        instances_employee_12m[employee].append(soma_value)
        #instances_employee_12m[employee].append(median_value)


## ---------- creating employees count dataset -----------------------
instances_Employee_BA_Type_12m=pd.DataFrame.from_dict(instances_employee_12m, orient='i

instances_Employee_BA_Type_12m ['$$_Conversion_Empl'] = instances_Employee_BA_Type_12m
instances_Employee_BA_Type_12m ['Count_Conversion_Empl'] = instances_Employee_BA_Type_1

# changing 12Months_Conversion value > BA max $$ conversion
# high conv. values only indicate the employee converted a fow opps and didn't register
conditions = [(instances_Employee_BA_Type_12m ['$$_Conversion_Empl'] > max_value_conver
              (instances_Employee_BA_Type_12m ['$$_Conversion_Empl'] <= max_value_conve
             ]

choices = [max_value_conversion, instances_Employee_BA_Type_12m ['$$_Conversion_Empl']]

instances_Employee_BA_Type_12m ['$$_Conversion_Empl'] = np.select(conditions, choices)

#changing 12Months_Conversion value > BA max count conversion
# high conv. values only indicate the employee converted a fow opps and didn't register

conditions = [(instances_Employee_BA_Type_12m ['Count_Conversion_Empl'] > max_count_con
              (instances_Employee_BA_Type_12m ['Count_Conversion_Empl'] <= max_count_co
             ]

choices = [max_count_conversion, instances_Employee_BA_Type_12m ['Count_Conversion_Empl


instances_Employee_BA_Type_12m ['Count_Conversion_Empl'] = np.select(conditions, choice


#display(instances_Employee_BA_Type_12m)
```

In [101…

```
print(instances_Employee_BA_Type_12m ['$$_Conversion_Empl'].max())
```

```
3.120689655172414
```

Checking 12 months employees' conversion rate by BU and Business Type

In [102…

```
object_matter = 'Employee-BU-Type'
linhas = np.unique(dataset [object_matter])
colunas = list(np.unique(dataset['Final_Status']))
# BA = np.unique(dataset['BA'])

instances_employee_12m = {}
```

```
col_name =[]

for a in linhas:
    instances_employee_12m [a] = []

for status in colunas:
    name = "1y_count_" + str(status)
    col_name.append(name)

    name = "1y_sum_$$_" + str(status)
    col_name.append(name)

    for employee in linhas:
        status_cluster = dataset[ (dataset['Final_Status'] == status) & (dataset[object
        soma_count = status_cluster.shape[0]
        soma_value = (sum(status_cluster['Exp. Sales Vol. in CHF']))/1000000
        #median_value = (status_cluster['Exp. Sales Vol. in CHF'].median())/1000000
        instances_employee_12m[employee].append(soma_count)
        instances_employee_12m[employee].append(soma_value)
        #instances_employee_12m[employee].append(median_value)


## ---------- creating employees count dataset -----------------------
instances_Employee_BU_Type_12m=pd.DataFrame.from_dict(instances_employee_12m, orient='i

instances_Employee_BU_Type_12m ['$$_Conversion_Empl'] = instances_Employee_BU_Type_12m
instances_Employee_BU_Type_12m ['Count_Conversion_Empl'] = instances_Employee_BU_Type_1

# changing 12Months_Conversion value > BA max $$ conversion
# high conv. values only indicate the employee converted a fow opps and didn't register
conditions = [(instances_Employee_BU_Type_12m ['$$_Conversion_Empl'] > max_value_conver
              (instances_Employee_BU_Type_12m ['$$_Conversion_Empl'] <= max_value_conve
             ]

choices = [max_value_conversion, instances_Employee_BU_Type_12m ['$$_Conversion_Empl']]

instances_Employee_BU_Type_12m ['$$_Conversion_Empl'] = np.select(conditions, choices)

#changing 12Months_Conversion value > BA max count conversion
# high conv. values only indicate the employee converted a fow opps and didn't register

conditions = [(instances_Employee_BU_Type_12m ['Count_Conversion_Empl'] > max_count_con
              (instances_Employee_BU_Type_12m ['Count_Conversion_Empl'] <= max_count_co
             ]

choices = [max_count_conversion, instances_Employee_BU_Type_12m ['Count_Conversion_Empl


instances_Employee_BU_Type_12m ['Count_Conversion_Empl'] = np.select(conditions, choice

#display(instances_Employee_BU_Type_12m)
```

Future Pipeline: active opps for CS, SMB and P&P

In [103…

```
# create dataset:
# opps for P&P, SMB and CS
# active opps with release date > Last month (to include opps not fully updated)
```

```
initial_analysis = date #date was inputed at the beging
delta = timedelta(days=30)
start = initial_analysis - delta

print('This analysis will start on', start)

dataset = som_df_modified[
                            (som_df_modified['PlanDate: Order Rel.'] >= start ) &
                            (som_df_modified['Final_Status'] == 'Active')
                                                   ]
dataset.shape
```

```
This analysis will start on 2022-08-03 00:00:00
```

Out[103…

```
(2894, 48)
```

Macro description of active opps value by BAs

In [104…

```
#Describing the pipeline by BAs & Opps Value

instances_describe = {}
BA_summary ={}

BA_summary = dataset['Exp. Sales Vol. in CHF'].describe()
BA_summary['Total'] = sum(dataset['Exp. Sales Vol. in CHF'])
for chave in BA_summary.keys():
        instances_describe[chave]=[]
        instances_describe[chave].append(BA_summary[chave])

#print(instances_describe)

for BA in np.unique(dataset['BA']):
    cluster_BA=dataset[(dataset['BA'] == BA)]
    BA_summary=cluster_BA['Exp. Sales Vol. in CHF'].describe()
    BA_summary['Total'] = sum(cluster_BA['Exp. Sales Vol. in CHF'])
    for chave in BA_summary.keys():
        #instances_describe[chave]=[]
        #print(chave)
        instances_describe[chave].append(BA_summary[chave])

coluna = ['BNAM']
coluna = coluna + (list(np.unique(np.unique(dataset['BA']))))

#print(coluna)

instances_describe=pd.DataFrame.from_dict(instances_describe, orient='index', columns=c

#display(instances_describe)

instances_describe=instances_describe.transpose()

#adding a date column
instances_describe ['Date']= pd.to_datetime(date, dayfirst=True)


#display(instances_describe)

#saving dataset
```

```
file_name ='%s_description_value_active_opps.csv'%tag
source_df = instances_describe
nome_index = "BA"
output_path(file_name, source_df, nome_index)


#saving dataset
#instances_describe.to_csv('%s_description_value_active_opps.csv'%tag, index=True, inde
```

Aggregating the count and value of opps by BAs & Business Type

In [105…

```
chaves = np.unique(dataset['BA - Business Type'])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

for i in chaves:
    BA_cluster = dataset[(dataset['BA - Business Type'] == i)]

    soma_count= BA_cluster.shape[0]

    soma_value= (sum(BA_cluster['Exp. Sales Vol. in CHF']))
    median_value = BA_cluster['Exp. Sales Vol. in CHF'].median()
    Q1_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.25)
    Q3_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.75)
    max_value = BA_cluster['Exp. Sales Vol. in CHF'].max()
    min_value = BA_cluster['Exp. Sales Vol. in CHF'].min()

    median_duration = BA_cluster['Opportunity Duration (No of Days)'].median()
    Q3_duration = BA_cluster['Opportunity Duration (No of Days)'].quantile(0.75)
    max_duration = BA_cluster['Opportunity Duration (No of Days)'].max()

    median_update = BA_cluster['Days since last update'].median()
    average_update = (BA_cluster['Days since last update'].mean()) #mean value of last
    Q3_update = BA_cluster['Days since last update'].quantile(0.75)
    max_update = BA_cluster['Days since last update'].max()

    instances_active[i].append(soma_count)

    instances_active[i].append(soma_value)
    instances_active[i].append(median_value)
    instances_active[i].append(Q1_value)
    instances_active[i].append(Q3_value)
    instances_active[i].append(max_value)
    instances_active[i].append(min_value)

    instances_active[i].append(median_duration)
    instances_active[i].append(Q3_duration)
    instances_active[i].append(max_duration)

    instances_active[i].append(median_update)
    instances_active[i].append(Q3_update)
    instances_active[i].append(max_update)

#print(instances_active)

# getting the main descriptions for BNAM - Business Type
for a in np.unique(dataset['Business Type']):
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]

#print(instances_active)

for a in np.unique(dataset['Business Type']):
    BA_cluster = dataset[(dataset['Business Type'] == a)]

    soma_count= BA_cluster.shape[0]

    soma_value= (sum(BA_cluster['Exp. Sales Vol. in CHF']))
    median_value = BA_cluster['Exp. Sales Vol. in CHF'].median()
    Q1_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.25)
    Q3_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.75)
    max_value = BA_cluster['Exp. Sales Vol. in CHF'].max()
    min_value = BA_cluster['Exp. Sales Vol. in CHF'].min()

    median_duration = BA_cluster['Opportunity Duration (No of Days)'].median()
    Q3_duration = BA_cluster['Opportunity Duration (No of Days)'].quantile(0.75)
    max_duration = BA_cluster['Opportunity Duration (No of Days)'].max()

    median_update = BA_cluster['Days since last update'].median()
    average_update = (BA_cluster['Days since last update'].mean()) #mean value of last
    Q3_update = BA_cluster['Days since last update'].quantile(0.75)
    max_update = BA_cluster['Days since last update'].max()

    instances_active['BNAM-%s'%a].append(soma_count)

    instances_active['BNAM-%s'%a].append(soma_value)
    instances_active['BNAM-%s'%a].append(median_value)
    instances_active['BNAM-%s'%a].append(Q1_value)
    instances_active['BNAM-%s'%a].append(Q3_value)
    instances_active['BNAM-%s'%a].append(max_value)
    instances_active['BNAM-%s'%a].append(min_value)

    instances_active['BNAM-%s'%a].append(median_duration)
    instances_active['BNAM-%s'%a].append(Q3_duration)
    instances_active['BNAM-%s'%a].append(max_duration)

    instances_active['BNAM-%s'%a].append(median_update)
    instances_active['BNAM-%s'%a].append(Q3_update)
    instances_active['BNAM-%s'%a].append(max_update)


# getting the main descriptions for BNAM

instances_active['BNAM-All']=[]

BA_cluster = dataset

soma_count= BA_cluster.shape[0]

soma_value= (sum(BA_cluster['Exp. Sales Vol. in CHF']))
```

```
median_value = BA_cluster['Exp. Sales Vol. in CHF'].median()
Q1_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.25)
Q3_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.75)
max_value = BA_cluster['Exp. Sales Vol. in CHF'].max()
min_value = BA_cluster['Exp. Sales Vol. in CHF'].min()

median_duration = BA_cluster['Opportunity Duration (No of Days)'].median()
Q3_duration = BA_cluster['Opportunity Duration (No of Days)'].quantile(0.75)
max_duration = BA_cluster['Opportunity Duration (No of Days)'].max()

median_update = BA_cluster['Days since last update'].median()
average_update = (BA_cluster['Days since last update'].mean()) #mean value of last upda
Q3_update = BA_cluster['Days since last update'].quantile(0.75)
max_update = BA_cluster['Days since last update'].max()

instances_active['BNAM-All'].append(soma_count)

instances_active['BNAM-All'].append(soma_value)
instances_active['BNAM-All'].append(median_value)
instances_active['BNAM-All'].append(Q1_value)
instances_active['BNAM-All'].append(Q3_value)
instances_active['BNAM-All'].append(max_value)
instances_active['BNAM-All'].append(min_value)

instances_active['BNAM-All'].append(median_duration)
instances_active['BNAM-All'].append(Q3_duration)
instances_active['BNAM-All'].append(max_duration)

instances_active['BNAM-All'].append(median_update)
instances_active['BNAM-All'].append(Q3_update)
instances_active['BNAM-All'].append(max_update)


#print(instances_active)




## ---------- creating opps. descriptions dataset -----------------------
instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns=[
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    '25% va
                                                                                    '75% va
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                    'Active
                                                                                       ]
                                        )

#display(instances_active)
```

```
In [106…
```

```
BAs_consolidated = pd.DataFrame()

BAs_consolidated = pd.merge(opps_BA_Type_1year, instances_active, left_index=True, righ

#display(BAs_consolidated)
```

In [107…

```
#Agregating the value of active opps by BAs-Business Type and Duration

chaves = np.unique(dataset['BA - Business Type'])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

# getting values for BA - Business Type
for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]


# getting values for BNAM - Business Type
for a in np.unique(dataset['Business Type']):
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]

# getting values for BNAM
instances_active['BNAM-All']=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Median update days of opps from %s'%(start/365) + ' to %s'%(end/365) +'
    coluna.append(col_label)
    duration_cluster = dataset[
                            (dataset['Opportunity Duration (No of Days)'] >= start) &
                            (dataset['Opportunity Duration (No of Days)'] < end)

                        ]

    #aggregating by BA - Business Type
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster['BA - Business Type'] == i)]
        median_update = BA_cluster['Days since last update'].median()
        instances_active[i].append(median_update)

    #aggregating by Business Type
    for a in np.unique(dataset['Business Type']):
        BA_cluster = duration_cluster[(duration_cluster['Business Type'] == a)]
        median_update = BA_cluster['Days since last update'].median()
        instances_active['BNAM-%s'%a].append(median_update)

    #aggregating by BNAM
    BA_cluster = duration_cluster
    median_update = BA_cluster['Days since last update'].median()
    instances_active['BNAM-All'].append(median_update)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)

BAs_consolidated = pd.merge(BAs_consolidated, instances_active, left_index=True, right_
```

In [108…

```
#Agregating the value of opps by BAs-Business Type and Exp. sales

chaves = np.unique(dataset['BA - Business Type'])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]


# getting the main descriptions for BNAM - Business Type
for a in np.unique(dataset['Business Type']):
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]
    instances_active['BNAM-%s'%a]=[]

instances_active['BNAM-All']=[]

coluna = []
start = 0

for dias in range (1,13):
    end = 365/2*dias
    #print(start,end)
    col_label= 'Opps value(million CFH) from %s'%(start/365) + ' to %s'%(end/365) +' du
    coluna.append(col_label)
    duration_cluster = dataset[
                              (dataset['Opportunity Duration (No of Days)'] >= start) &
                              (dataset['Opportunity Duration (No of Days)'] < end)
                         ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster['BA - Business Type'] == i)]
        soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000)
        instances_active[i].append(soma_value)

    for a in np.unique(dataset['Business Type']):
        BA_cluster = duration_cluster[(duration_cluster['Business Type'] == a)]
        soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000)
        instances_active['BNAM-%s'%a].append(soma_value)

    BA_cluster = duration_cluster
    soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000000)
    instances_active['BNAM-All'].append(soma_value)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)

BAs_consolidated = pd.merge(BAs_consolidated, instances_active, left_index=True, right_
```

In [ ]:

```

```

In [109…

```
#adding a date column
BAs_consolidated ['Date']= date

#converting index in column
BAs_consolidated.reset_index(inplace=True)
BAs_consolidated = BAs_consolidated.rename(columns = {'index':'BA-Business Type'})

#splitting BU - Business Type
BAs_consolidated['aux'] = BAs_consolidated ['BA-Business Type']
BAs_consolidated [['BA', 'Business Type']]= BAs_consolidated.aux.str.split("-",expand=T
BAs_consolidated.drop('aux', inplace=True, axis=1)

BAs_consolidated ['Date']= date

#display(BAs_consolidated)

#saving dataset
file_name ='%s_BA_Business_Type_overview.csv'%tag
source_df = BAs_consolidated
nome_index = "#"
output_path(file_name, source_df, nome_index)
```

Aggregating the count and value of opps by BUs & Business Type

In [110…

```
object_matter = 'BU - Business Type'
chaves = np.unique(dataset[object_matter])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

for i in chaves:
    BA_cluster = dataset[(dataset[object_matter] == i)]

    soma_count= BA_cluster.shape[0]

    soma_value= (sum(BA_cluster['Exp. Sales Vol. in CHF']))
    median_value = BA_cluster['Exp. Sales Vol. in CHF'].median()
    Q1_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.25)
    Q3_value = BA_cluster['Exp. Sales Vol. in CHF'].quantile(0.75)
    max_value = BA_cluster['Exp. Sales Vol. in CHF'].max()
    min_value = BA_cluster['Exp. Sales Vol. in CHF'].min()

    median_duration = BA_cluster['Opportunity Duration (No of Days)'].median()
    Q3_duration = BA_cluster['Opportunity Duration (No of Days)'].quantile(0.75)
    max_duration = BA_cluster['Opportunity Duration (No of Days)'].max()

    median_update = BA_cluster['Days since last update'].median()
    average_update = (BA_cluster['Days since last update'].mean()) #mean value of last
    Q3_update = BA_cluster['Days since last update'].quantile(0.75)
    max_update = BA_cluster['Days since last update'].max()

    instances_active[i].append(soma_count)

    instances_active[i].append(soma_value)
    instances_active[i].append(median_value)
    instances_active[i].append(Q1_value)
    instances_active[i].append(Q3_value)
    instances_active[i].append(max_value)
    instances_active[i].append(min_value)

    instances_active[i].append(median_duration)
    instances_active[i].append(Q3_duration)
    instances_active[i].append(max_duration)

    instances_active[i].append(median_update)
    instances_active[i].append(Q3_update)
    instances_active[i].append(max_update)



## ---------- creating opps. descriptions dataset -----------------------
instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns=[
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 '25% va
                                                                                 '75% va
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                 'Active
                                                                                     ]
                                              )

#print(instances_active)
#display(instances_active)




#---------------------------------------------------------------------------------
#merging BU datasets
```

```
BUs_consolidated = pd.DataFrame()

BUs_consolidated = pd.merge(opps_BU_Type_1year, instances_active, left_index=True, righ



#display(BUs_consolidated)

#-----------------------------------------------------------------------------------
#Agregating the value of opps by BUs-Business Type and Duration

object_matter = 'BU - Business Type'
#chaves = np.unique(dataset[object_matter])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []

instances_active = {}

# getting values for BA - Business Type
for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Median update days of opps from %s'%(start/365) + ' to %s'%(end/365) +'
    coluna.append(col_label)
    duration_cluster = dataset[
                         (dataset['Opportunity Duration (No of Days)'] >= start) &
                         (dataset['Opportunity Duration (No of Days)'] < end)
                     ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        median_update = BA_cluster['Days since last update'].median()
        instances_active[i].append(median_update)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)
#print(instances_active)
#print(coluna)


#-----------------------------------------------------------------
BUs_consolidated = pd.merge(BUs_consolidated, instances_active, left_index=True, right_

#-----------------------------------------------------
#Agregating the value of opps by BUs-Business Type and Exp. sales

#chaves = np.unique(dataset[object_matter])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []
```

```
instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Opps value(CFH) from %s'%(start/365) + ' to %s'%(end/365) +' duration y
    coluna.append(col_label)
    duration_cluster = dataset[
                             (dataset['Opportunity Duration (No of Days)'] >= start) &
                             (dataset['Opportunity Duration (No of Days)'] < end)
                        ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF']))
        instances_active[i].append(soma_value)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)

#----------------------------------------------------------------
#Final aggregations
BUs_consolidated = pd.merge(BUs_consolidated, instances_active, left_index=True, right_
#print(instances_active)
#print(coluna)


#adding a date column
BUs_consolidated ['Date']= date

#converting index in column
BUs_consolidated.reset_index(inplace=True)
BUs_consolidated = BUs_consolidated.rename(columns = {'index':object_matter})

BUs_consolidated ['Date']= date

#display(BUs_consolidated)

#saving dataset
file_name ='%s_BU_Business_Type_overview.csv'%tag
source_df = BUs_consolidated
nome_index = "#"
output_path(file_name, source_df, nome_index)
```

Aggregating the count, value, duration and updated info for Employees - BA - Business Type

In [111…

```
#Aggregating the count and value of opps by  Employees

object_matter = 'Employee-BA-Type'
chaves = np.unique(dataset[object_matter])
instances_summary = {}

for a in chaves:
    instances_summary[a]= []

for i in chaves:
    BA_cluster = dataset[(dataset[object_matter] == i)] #sorting each employee responsi
    soma_count = BA_cluster.shape[0]
    soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF']/1000000))
    median_value = (BA_cluster['Exp. Sales Vol. in CHF'].median()/1000000)
    average_value = (BA_cluster['Exp. Sales Vol. in CHF'].mean()/1000000)
    min_value = (BA_cluster['Exp. Sales Vol. in CHF'].min()/1000000)
    max_value = (BA_cluster['Exp. Sales Vol. in CHF'].max()/1000000)

    median_age = (BA_cluster['Opportunity Duration (No of Days)'].median())
    average_age = (BA_cluster['Opportunity Duration (No of Days)'].mean())
    max_age = (BA_cluster['Opportunity Duration (No of Days)'].max())

    median_update = (BA_cluster['Days since last update'].median()) #median value of la
    average_update = (BA_cluster['Days since last update'].mean()) #mean value of last
    max_update = (BA_cluster['Days since last update'].max())

    instances_summary[i].append(soma_count) #Value
    instances_summary[i].append(soma_value) #Value
    instances_summary[i].append(median_value) #Value
    instances_summary[i].append(average_value) #Value
    instances_summary[i].append(min_value) #Value
    instances_summary[i].append(max_value) #Value

    instances_summary[i].append(median_age) #Value
    instances_summary[i].append(average_age) #Value
    instances_summary[i].append(max_age) #Value

    instances_summary[i].append(median_update) #Value
    instances_summary[i].append(average_update) #Value
    instances_summary[i].append(max_update) #Value

## ---------- creating employees count dataset -----------------------

description = 'active_opps'
instances_summary = pd.DataFrame.from_dict(
    instances_summary,
    orient='index',
    columns=['count_%s'%description,
             'Value_$$_%s'%description,
             'Median_$$_%s'%description,
             'Average_$$_%s'%description,
             'Min_$$_%s'%description,
             'Max_$$_%s'%description,
             'Median_Age_%s'%description,
             'Avrg_Age_%s'%description,
             'Max_Age_(years)%s'%description,
             'Median Last Update_%s'%description,
             'Mean Last Update_%s'%description,
             'Max days since last update_%s'%description
                ])
```

```
# Merging past 12months with active opps analysis
instances_summary = pd.merge(instances_Employee_BA_Type_12m,instances_summary,left_inde

instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Median update days of opps from %s'%(start/365) + ' to %s'%(end/365) +'
    coluna.append(col_label)
    duration_cluster = dataset[
                        (dataset['Opportunity Duration (No of Days)'] >= start) &
                        (dataset['Opportunity Duration (No of Days)'] < end)
                    ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        median_update = BA_cluster['Days since last update'].median()
        instances_active[i].append(median_update)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)
#print(instances_active)
#print(coluna)


#--------------------------------------------------------------------
# Merging summary with update info
instances_summary = pd.merge(instances_summary,instances_active, left_index=True, right


#-------------------------------------------------------
#Agregating the value of opps by BUs-Business Type and Exp. sales

#chaves = np.unique(dataset[object_matter])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []
instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Opps value(kCFH) from %s'%(start/365) + ' to %s'%(end/365) +' duration
    coluna.append(col_label)
    duration_cluster = dataset[
                            (dataset['Opportunity Duration (No of Days)'] >= start) &
                            (dataset['Opportunity Duration (No of Days)'] < end)
                        ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF'])/1000)
        instances_active[i].append(soma_value)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

# Merging summary with update info
instances_summary = pd.merge(instances_summary,instances_active, left_index=True, right


#display(instances_summary)

# Transforming the index into a column
instances_summary.reset_index(inplace=True)
instances_summary = instances_summary.rename(columns = {'index':'EmployeeBAType'})

#display(instances_summary)

# Spliting the new column
instances_summary[['Employee','BA','Business Type']] = instances_summary.EmployeeBAType

instances_summary ['BA-Business Type'] = instances_summary['BA'] + '-' + instances_summ

wons_duration = BAs_consolidated.iloc[:,[0,11,12]]

instances_summary = pd.merge(instances_summary, wons_duration, on = 'BA-Business Type',


# Adding a date column
instances_summary ['Date']= date

#display(instances_summary)

#saving dataset
file_name ='%s_'%tag + '%s_overview.csv'%object_matter
source_df = instances_summary
nome_index = "#"
output_path(file_name, source_df, nome_index)
```

Aggregating the count, value, duration and updated info for Employees - BU - Business Type

In [112…

```
#Aggregating the count and value of opps by  Employees

object_matter = 'Employee-BU-Type'
chaves = np.unique(dataset[object_matter])
```

```
instances_summary = {}

for a in chaves:
    instances_summary[a]= []

for i in chaves:
    BA_cluster = dataset[(dataset[object_matter] == i)] #sorting each employee responsi
    soma_count = BA_cluster.shape[0]
    soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF']))
    median_value = (BA_cluster['Exp. Sales Vol. in CHF'].median())
    average_value = (BA_cluster['Exp. Sales Vol. in CHF'].mean())
    min_value = (BA_cluster['Exp. Sales Vol. in CHF'].min())
    max_value = (BA_cluster['Exp. Sales Vol. in CHF'].max())

    median_age = (BA_cluster['Opportunity Duration (No of Days)'].median())
    average_age = (BA_cluster['Opportunity Duration (No of Days)'].mean())
    max_age = (BA_cluster['Opportunity Duration (No of Days)'].max())

    median_update = (BA_cluster['Days since last update'].median()) #median value of la
    average_update = (BA_cluster['Days since last update'].mean()) #mean value of last
    max_update = (BA_cluster['Days since last update'].max())

    instances_summary[i].append(soma_count) #Value
    instances_summary[i].append(soma_value) #Value
    instances_summary[i].append(median_value) #Value
    instances_summary[i].append(average_value) #Value
    instances_summary[i].append(min_value) #Value
    instances_summary[i].append(max_value) #Value

    instances_summary[i].append(median_age) #Value
    instances_summary[i].append(average_age) #Value
    instances_summary[i].append(max_age) #Value

    instances_summary[i].append(median_update) #Value
    instances_summary[i].append(average_update) #Value
    instances_summary[i].append(max_update) #Value

## ---------- creating employees count dataset -----------------------

description = 'active_opps'
instances_summary = pd.DataFrame.from_dict(
    instances_summary,
    orient='index',
    columns=['count_%s'%description,
             'Value_$$_%s'%description,
             'Median_$$_%s'%description,
             'Average_$$_%s'%description,
             'Min_$$_%s'%description,
             'Max_$$_%s'%description,
             'Median_Age_%s'%description,
             'Avrg_Age_%s'%description,
             'Max_Age_%s'%description,
             'Median Last Update_%s'%description,
             'Mean Last Update_%s'%description,
             'Max days since last update_%s'%description
                ])

# Merging past 12months with active opps analysis
instances_summary = pd.merge(instances_Employee_BU_Type_12m,instances_summary,left_inde
instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Median update days of opps from %s'%(start/365) + ' to %s'%(end/365) +'
    coluna.append(col_label)
    duration_cluster = dataset[
                         (dataset['Opportunity Duration (No of Days)'] >= start) &
                         (dataset['Opportunity Duration (No of Days)'] < end)
                    ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        median_update = BA_cluster['Days since last update'].median()
        instances_active[i].append(median_update)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

#display(instances_active)
#print(instances_active)
#print(coluna)



#--------------------------------------------------------------------
# Merging summary with update info
instances_summary = pd.merge(instances_summary,instances_active, left_index=True, right


#-------------------------------------------------------
#Agregating the value of opps by BUs-Business Type and Exp. sales

#chaves = np.unique(dataset[object_matter])
#colunas = list(np.unique(dataset['Final_Status']))
colunas = []
instances_active = {}

for a in chaves:
    instances_active[a] = []
    #instances_value[a] = []
    #instances_duration[a]=[]

coluna = []
start = 0

for dias in range (1,13):
    end = (365/2)*dias
    #print(start,end)
    col_label= 'Opps value(CFH) from %s'%(start/365) + ' to %s'%(end/365) +' duration y
    coluna.append(col_label)
    duration_cluster = dataset[
                                (dataset['Opportunity Duration (No of Days)'] >= start) &
                                (dataset['Opportunity Duration (No of Days)'] < end)
                            ]
    for i in chaves:
        BA_cluster = duration_cluster[(duration_cluster[object_matter] == i)]
        soma_value = (sum(BA_cluster['Exp. Sales Vol. in CHF']))
        instances_active[i].append(soma_value)

    start=end


instances_active=pd.DataFrame.from_dict(instances_active, orient='index', columns = col

# Merging summary with update info
instances_summary = pd.merge(instances_summary,instances_active, left_index=True, right


#display(instances_summary)

# Transforming the index into a column
instances_summary.reset_index(inplace=True)
instances_summary = instances_summary.rename(columns = {'index':'EmployeeBUType'})

#display(instances_summary)

# Spliting the new column
instances_summary[['Employee','BU','Business Type']] = instances_summary.EmployeeBUType
instances_summary = instances_summary.rename(columns = {'EmployeeBUType':object_matter}

# Adding a date column
instances_summary ['Date']= date

#display(instances_summary)

#saving dataset
file_name ='%s_'%tag + '%s_overview.csv'%object_matter
source_df = instances_summary
nome_index = "#"
output_path(file_name, source_df, nome_index)
```

Generating the SOM for the Sales Leadership Report

In [113…

```
## Creating Dataset for Sales Leadership Report

# create dataset:
# opps for all Business Types (CS, P&P and SMB
# active opps with release date > Last month (to include opps not fully updated)

initial_analysis = date #date was inputed at the begining
delta = timedelta(days=30)
start = initial_analysis - delta

print('This analysis will start on', start)

dataset = som_df_modified[
                            (som_df_modified['PlanDate: Order Rel.'] >= start ) &
```

```
                                            (som_df_modified['Final_Status'] == 'Active')
                                                                ]



# Merging instances_summary with instances_describe_median by BA
dataset = pd.merge(dataset,BUs_consolidated, on ='BU - Business Type', left_index=False
dataset = pd.merge(dataset,instances_summary, on = 'Employee-BU-Type', left_index=False


#saving dataset
file_name = '%s_SOM_all_active.csv'%tag
source_df = dataset
nome_index = "#"
output_path(file_name, source_df, nome_index)

#saving dataset
#dataset.to_csv('%s_SOM_all_active.csv'%tag,  encoding='utf-8', index=False)

dataset.shape
```

```
This analysis will start on 2022-08-03 00:00:00
```

Out[113…
```
(2894, 144)
```

In [114…
```
import datetime

print("Last executed on:", datetime.datetime.now())
```

```
Last executed on: 2022-09-02 21:03:56.173746
```

In [ ]: