Vue model point ASAC

October 8, 2020

In [2]:
```
import os,sys,time,pyspark,shutil
import pandas as pd
os.environ['PYSPARK_PYTHON'] = '/misc/anaconda3/envs/py35spark/bin/python'
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql import SQLContext
from pyspark.sql import SparkSession
```

In [3]:
```
conf = pyspark.SparkConf()
conf.setMaster("spark://azfr2-spark-production-mstr-master.service.dsp.allianz:7077") \
    .set("spark.cores.max","8").set("spark.executor.memory", "6g").set("spark.executor.c
    .set("spark.driver.memory", "6g").setAppName("Jupyter_PROD_larcher_Model_Point")
sc = pyspark.SparkContext(conf=conf)

spark = SparkSession.builder.getOrCreate()
sqlContext = SQLContext(sc)
```

In [3]:
```
def get_max_data_date_partition(parquet):
    global sqlContext
    df = sqlContext.sql("SELECT MAX(DATA_DATE_PARTITION) AS MAX FROM parquet.`"+parquet+
    return (df.first().MAX)
```

In [4]:
```
read_parquet_SAVPROD_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVPROD.parquet/"
read_parquet_SAVCNTIS_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVCNTIS.parquet/"
read_parquet_SAVGARSO_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVGARSO.parquet/"
read_parquet_SAVGAR_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVGAR.parquet/"
read_parquet_SAVTPGGA_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVTPGGA.parquet/"
read_parquet_SATRSR_SA01 = "/data/dropbox/larcher/ASAC/SA01_SATRSR.parquet/"
read_parquet_SAVPEROL_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVPEROL.parquet/"
read_parquet_SAVPPHY_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVPPHY.parquet/"
read_parquet_SATQUAL_SA01 = "/data/dropbox/larcher/ASAC/SA01_SATQUAL.parquet/"
read_parquet_SAVOPE_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVOPE.parquet/"
read_parquet_SAVMPA_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVMPA.parquet/"
read_parquet_SATTYOPE_SA01 = "/data/dropbox/larcher/ASAC/SA01_SATTYOPE.parquet/"
read_parquet_SAVGARMP_SA01 = "/data/dropbox/larcher/ASAC/SA01_SAVGARMP.parquet/"
#read_parquet_SA03_inventaire = "/data/prod_env/data/parsed_data/SA03/VC/SA03_EXERN.parq
read_parquet_SA03_inventaire = "/data/dropbox/larcher/ASAC/VC/SA03_EXERN.parquet"
```

```
In [5]: def traitement_SAVPROD(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVPROD_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_p
            df_SA01_SAVPROD = sqlContext.read.parquet(read_parquet_SAVPROD_SA01)
            df_SA01_SAVPROD = df_SA01_SAVPROD.select("CD_FIRME_PROD","CD_REGIME_PROD").filter("C

            return df_SA01_SAVPROD

        def traitement_SAVCNTIS(parquet):

            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVCNTIS_SA01 = parquet + "DATA_DATE_PARTITION=" + str(last_data_date_p
            df_SA01_SAVCNTIS = sqlContext.read.parquet(read_parquet_SAVCNTIS_SA01)
            df_SA01_SAVCNTIS = df_SA01_SAVCNTIS.select("CD_FIRM_PROD_CNTIS","NU_CNT_IND_CNTIS","
                                               .withColumn("TYP_INTERMEDIAIRE_TRANSCO",when(df_S
                                               .withColumn("DUREE_RENTE",lit('')) \
                                               .withColumn("TYPE_GESTION",lit("LIBRE")) \
                                               .withColumn("SUPPORT_INVESTISSEMENT",lit("EUROS")
                                               .withColumn("TYPE_SUPPORT",lit("EUROS")) \
        #                                       .filter("NU_CNT_IND_CNTIS = '120676'")

            return df_SA01_SAVCNTIS

        def traitement_SAVGARSO(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVGARSO_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_
            df_SA01_SAVGARSO = sqlContext.read.parquet(read_parquet_SAVGARSO_SA01)
            df_SA01_SAVGARSO = df_SA01_SAVGARSO.select("NU_CNT_IND_GARSO","NU_ORD_GAS_GARSO","NU

            return df_SA01_SAVGARSO

        def traitement_SAVGAR(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVGAR_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_pa
            df_SA01_SAVGAR = sqlContext.read.parquet(read_parquet_SAVGAR_SA01)
            df_SA01_SAVGAR = df_SA01_SAVGAR.select("NU_GAR","LBC_GAR","LB_GAR","CD_RSQ_GAR","CD_
                                           .withColumn("TARIF",lit(""))

            return df_SA01_SAVGAR

        def traitement_SAVTPGGA(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVTPGGA_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_
            df_SA01_SAVTPGGA = sqlContext.read.parquet(read_parquet_SAVTPGGA_SA01)
            df_SA01_SAVTPGGA = df_SA01_SAVTPGGA.select("NU_GAR_TPGGA","CD_TPG_TPGGA")

            return df_SA01_SAVTPGGA
```

```
def traitement_SATRSR(parquet):
    last_data_date_partition = get_max_data_date_partition(parquet)
    read_parquet_SATRSR_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_pa
    df_SA01_SATRSR = sqlContext.read.parquet(read_parquet_SATRSR_SA01)
    df_SA01_SATRSR = df_SA01_SATRSR.select("CD_RSQ","CD_SRSQ","LB_RSQ") \
                                   .withColumn("TAUX_INTERET_TECHNIQUE",lit("")) \
                                   .withColumn("COEFFICIENT_GARANTIE_PLANCHER",lit(""))

    return df_SA01_SATRSR


def traitement_SAVPEROL(parquet):
    last_data_date_partition = get_max_data_date_partition(parquet)
    read_parquet_SAVPEROL_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_
    df_SA01_SAVPEROL = sqlContext.read.parquet(read_parquet_SAVPEROL_SA01)
    df_SA01_SAVPEROL = df_SA01_SAVPEROL.select("NU_CNT_IND_PEROL","NU_PERS_PEROL","NU_TY
                                       .withColumn("NU_PERS_PEROL2",lit("")) \
                                       .withColumn("NU_TYP_ROLE_PEROL2",lit(""))

    return df_SA01_SAVPEROL

def traitement_SAVPPHY(parquet):
    last_data_date_partition = get_max_data_date_partition(parquet)
    read_parquet_SAVPPHY_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_p
    df_SA01_SAVPPHY = sqlContext.read.parquet(read_parquet_SAVPPHY_SA01)
    df_SA01_SAVPPHY = df_SA01_SAVPPHY.select("NU_PERS_PHY","DT_NAISS_PHY","CD_QUAL_PHY")
                                     .withColumn("DT_NAISS_PHY2",lit(""))

    return df_SA01_SAVPPHY

def traitement_SATQUAL(parquet):
    last_data_date_partition = get_max_data_date_partition(parquet)
    read_parquet_SATQUAL_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_p
    df_SA01_SATQUAL = sqlContext.read.parquet(read_parquet_SATQUAL_SA01)
    df_SA01_SATQUAL = df_SA01_SATQUAL.select("CD_QUAL","CD_SEX_QUAL") \
                                     .withColumn("CD_SEX_QUAL_TRANSCO",when(df_SA01_SATQ
                                     .withColumn("CD_SEX_QUAL2",lit("")) \
                                     .withColumn("CD_SEX_QUAL_TRANSCO2",lit(""))


    return df_SA01_SATQUAL

def traitement_SAVOPE(parquet):
    last_data_date_partition = get_max_data_date_partition(parquet)
    read_parquet_SAVOPE_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_pa
    df_SA01_SAVOPE = sqlContext.read.parquet(read_parquet_SAVOPE_SA01)
    df_SA01_SAVOPE = df_SA01_SAVOPE.select("NU_OPE","NU_CNT_IND_OPE","NU_ORD_GAS_OPE","D
                                .withColumn("DATE_SITUATION",lit(last_data_date_partition
```

```
                                        .withColumn("FEEDER",lit("ASAC_SA01")) \
                                        .withColumn("TYPE_PRIME",lit('EUR')) \
                                        .withColumn("COMMISSION_GESTION",lit('')) \
                                        .withColumn("TYPE_COMMISSION_GESTION",lit('EUR')) \
                                        .withColumn("CHARGEMENT_GESTION",lit('')) \
                                        .withColumn("TYPE_CHARGEMENT_GESTION",lit('EUR')) \
                                        .withColumn("COMMISSION_PRODUIT_FINANCIER",lit("")) \
                                        .withColumn("TYPE_COMMISSION_PRODUIT_FINANCIER",lit('EUR'
                                        .withColumn("TOP_PRIMES_ARRERAGE",lit('PRIMES')) \

            return df_SA01_SAVOPE

        def traitement_SAVMPA(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVMPA_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_pa
            df_SA01_SAVMPA = sqlContext.read.parquet(read_parquet_SAVMPA_SA01)
            df_SA01_SAVMPA = df_SA01_SAVMPA.select("PERIO_MPA","NU_CNT_IND_MPA","DT_DEB_EFF_MPA"

            return df_SA01_SAVMPA

        def traitement_SATTYOPE(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SATTYOPE_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_
            df_SA01_SATTYOPE = sqlContext.read.parquet(read_parquet_SATTYOPE_SA01)
            df_SA01_SATTYOPE = df_SA01_SATTYOPE.select("NU_TYP_OPERATION","CD_TYP_OPERATION","LB
            return df_SA01_SATTYOPE

        def traitement_SAVGARMP(parquet):
            last_data_date_partition = get_max_data_date_partition(parquet)
            read_parquet_SAVGARMP_SA01 = parquet + "DATA_DATE_PARTITION=" +  str(last_data_date_
            df_SA01_SAVGARMP = sqlContext.read.parquet(read_parquet_SAVGARMP_SA01)
            df_SA01_SAVGARMP = df_SA01_SAVGARMP.select("NU_CNT_IND_GARMP","NU_ORD_GAS_GARMP","NU
            return df_SA01_SAVGARMP

        def traitement_inventaire_SA03(parquet):
            df_parquet_SA03_inventaire = sqlContext.read.parquet(parquet)
            return df_parquet_SA03_inventaire

In [6]: df_SA01_SAVPROD = traitement_SAVPROD(read_parquet_SAVPROD_SA01)
        df_SA01_SAVCNTIS = traitement_SAVCNTIS(read_parquet_SAVCNTIS_SA01)
        df_SA01_SAVGARSO = traitement_SAVGARSO(read_parquet_SAVGARSO_SA01)
        df_SA01_SAVGAR = traitement_SAVGAR(read_parquet_SAVGAR_SA01)
        df_SA01_SAVTPGGA = traitement_SAVTPGGA(read_parquet_SAVTPGGA_SA01)
        df_SA01_SATRSR = traitement_SATRSR(read_parquet_SATRSR_SA01)
        df_SA01_SAVPEROL = traitement_SAVPEROL(read_parquet_SAVPEROL_SA01)
        df_SA01_SAVPPHY = traitement_SAVPPHY(read_parquet_SAVPPHY_SA01)
        df_SA01_SATQUAL = traitement_SATQUAL(read_parquet_SATQUAL_SA01)
        df_SA01_SAVOPE = traitement_SAVOPE(read_parquet_SAVOPE_SA01)
```

```
        df_SA01_SAVMPA = traitement_SAVMPA(read_parquet_SAVMPA_SA01)
        df_SA01_SATTYOPE = traitement_SATTYOPE(read_parquet_SATTYOPE_SA01)
        df_SA01_SAVGARMP = traitement_SAVGARMP(read_parquet_SAVGARMP_SA01)
        df_parquet_SA03_inventaire = traitement_inventaire_SA03(read_parquet_SA03_inventaire)

In [ ]: # df_SA01_SAVMPA.printSchema()

In [ ]: # df_SA01_SAVMPA.distinct().filter("NU_CNT_IND_MPA = '50698'").orderBy("DT_DEB_EFF_MPA")

In [7]: df_enrichissement = df_SA01_SAVPROD.join(df_SA01_SAVCNTIS, df_SA01_SAVCNTIS["CD_FIRM_PRO
                                            .drop("CD_FIRM_PROD_CNTIS")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVGARSO, df_enrichissement["NU_CNT_I
                                              .drop("NU_CNT_IND_GARSO")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVGAR, df_enrichissement["NU_GAR_GAR
                                              .drop("NU_GAR")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVTPGGA, df_enrichissement["NU_GAR_G
                                              .drop("NU_GAR_TPGGA")

        df_enrichissement = df_enrichissement.join(df_SA01_SATRSR, (df_enrichissement["CD_RSQ_GA
                                              .drop("CD_RSQ_GAR")\
                                              .drop("CD_SRSQ_GAR")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVPEROL, df_enrichissement["NU_CNT_I
                                              .drop("NU_CNT_IND_PEROL")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVPPHY, df_enrichissement["NU_PERS_P
                                              .drop("NU_PERS_PHY")

        df_enrichissement = df_enrichissement.join(df_SA01_SATQUAL, df_enrichissement["CD_QUAL_P
                                              .drop("CD_QUAL") \
                                              .drop("CD_QUAL_PHY")

        df_enrichissement_for_inventaire  = df_enrichissement

        df_enrichissement = df_enrichissement.join(df_SA01_SAVOPE, (df_enrichissement["NU_CNT_IN
                                              .drop("NU_ORD_GAS_GARSO")

        df_enrichissement = df_enrichissement.join(df_SA01_SAVGARMP, (df_enrichissement["NU_CNT_
                                              .drop("NU_CNT_IND_OPE") \
        #                                      .drop("NU_ORD_GAS_OPE") \

        df_enrichissement = df_enrichissement.join(df_SA01_SATTYOPE, df_enrichissement["NU_TYP_O

        df_enrichissement = df_enrichissement.join(df_SA01_SAVMPA, (df_enrichissement["NU_CNT_IN
```

```
        df_enrichissement = df_enrichissement.filter(col("NU_TYP_OPERATI_OPE").isin([1,2,5,13,21
                                                                                    .where("DT_REGL_OPE <> '
                                                                                    .where("DT_ANNULATION_OP


        df_enrichissement = df_enrichissement.withColumn("PERIO_MPA",when(df_enrichissement["NU_
        #                                     .withColumn("PERIO_MPA_TRANSCO",when(df_enrichissem

        df_enrichissement = df_enrichissement.withColumn("PERIO_MPA_2",when(df_enrichissement["D
                                              #.drop("PERIO_MPA")
        df_enrichissement = df_enrichissement.withColumn("PERIO_MPA_TRANSCO",when(df_enrichissem
```

```
In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS").filter("PERIO_MPA_2 = 'A'").distinct().sh

In [ ]: #df_enrichissement = df_enrichissement.filter("NU_CNT_IND_CNTIS = '135275'")

In [ ]: # df_enrichissement = df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825')
        # df_enrichissement = df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2'

In [ ]: # df_enrichissement.show(100)

In [ ]: # df_enrichissement = df_enrichissement.drop(df_enrichissement[df_enrichissement["PERIO_
        # #.drop_duplicates(subset=['NU_OPE','DT_ENR_OPE'])

In [ ]: # df_enrichissement.show(100)
        # df_enrichissement.count()

In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2","D

In [ ]:

In [8]: import pyspark.sql.functions as f
        from pyspark.sql import Window

In [9]: df_enrichissement_mpa = df_enrichissement.where(df_enrichissement["DT_ENR_OPE"].between(
        #df_enrichissement_mpa.show(20,False)

In [ ]: # df_enrichissement_mpa.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2

In [10]: df_enrichissement_mpa = df_enrichissement_mpa.dropDuplicates(subset=['NU_CNT_IND_CNTIS'

In [ ]: # df_enrichissement_mpa_not_between = df_enrichissement.where(df_enrichissement["DT_ENR_

In [ ]: # df_enrichissement_mpa_not_between.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT

In [ ]: # df_enrichissement_mpa.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE","D
```

```
In [11]: df_enrichissement = df_enrichissement.filter("PERIO_MPA_2 =''")
         #df_enrichissement = df_enrichissement.filter(df_enrichissement["DT_ENR_OPE"].between(d
         #df_enrichissement.count()

In [ ]:

In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2","D

In [ ]: # df_enrichissement.count()

In [ ]: #df_enrichissement = df_enrichissement.where(~df_enrichissement["DT_ENR_OPE"].between(df

In [ ]: # df_enrichissement.count()

In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2","D

In [12]: df_enrichissement = df_enrichissement.dropDuplicates(subset=['NU_CNT_IND_CNTIS','NU_OPE

In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2","D

In [ ]: # df_enrichissement.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE","DT_DE

In [ ]: # print(df_enrichissement_mpa.filter('NU_CNT_IND_CNTIS = 23825').count())
         # print(df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825').count())

In [ ]: #df_enrichissement_mpa.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_ENR_OPE','DT

In [ ]: # df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_ENR_OPE','DT_DE

In [ ]: # w = Window.partitionBy('NU_CNT_IND_CNTIS', 'NU_OPE', 'DT_ENR_OPE')
         # df_enrichissement_mpavide = df_enrichissement.select('*', f.count('NU_CNT_IND_CNTIS').
         #     .where('dupeCount > 1')\
         #     .drop('dupeCount')\
         #     #.show(100)
         #df_enrichissement_mpavide.filter('NU_CNT_IND_CNTIS = 23825').count()

         #df_enrichissement_mpavide = df_enrichissement_mpavide.dropDuplicates(subset=['NU_OPE','
         #df_enrichissement.show(100)

In [ ]: # df_enrichissement_mpavide.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE

In [ ]: # df_enrichissement_mpavide_drop_duplicates = df_enrichissement_mpavide.dropDuplicates(s

In [ ]: # df_enrichissement_mpavide_drop_duplicates.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MP

In [ ]: # df_enrichissement_mpavide_drop_duplicates.filter('NU_CNT_IND_CNTIS = 23825').count()

In [ ]: #df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_ENR_OPE','DT_DEB

In [ ]: #df_enrichissement.count()
```

```
In [ ]: # print(df_enrichissement_mpa.filter('NU_CNT_IND_CNTIS = 23825').count())
        # print(df_enrichissement_mpavide_drop_duplicates.filter('NU_CNT_IND_CNTIS = 23825').cou

In [13]: df_enrichissement_final = df_enrichissement.union(df_enrichissement_mpa)
         df_enrichissement_final.createOrReplaceTempView('df_enrichissement_final')
         #df_enrichissement_final.count()

In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA

In [14]: w = Window.partitionBy('NU_CNT_IND_CNTIS','NU_OPE','NU_ORD_GAS_OPE', 'DT_ENR_OPE')
         df_enrichissement_uniques_rows = df_enrichissement_final.select('*', f.count('NU_CNT_IN
             .where('dupeCount = 1')\
             .drop('dupeCount')\
         #    .show(100)
         #df_enrichissement_final.filter('NU_CNT_IND_CNTIS = 23825').count()

In [ ]: # df_enrichissement_uniques_rows.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PE

In [ ]:

In [ ]: # query = """
        #         SELECT * FROM df_enrichissement_final
        #         WHERE NU_CNT_IND_CNTIS IN ( SELECT NU_CNT_IND_CNTIS FROM df_enrichissement_fin
        #         AND NU_OPE IN ( SELECT NU_OPE FROM df_enrichissement_final GROUP BY NU_OPE HAV
        #         AND DT_ENR_OPE IN ( SELECT DT_ENR_OPE FROM df_enrichissement_final GROUP BY DT
        #         AND NU_ORD_GAS_OPE IN ( SELECT NU_ORD_GAS_OPE FROM df_enrichissement_final GRO

        #         """
        # df_df_enrichissement_final = sqlContext.sql(query)

In [ ]: # query = """
        #   select distinct NU_CNT_IND_CNTIS, NU_OPE,DT_ENR_OPE,NU_ORD_GAS_OPE, PERIO_MPA_2, DT_
        #   from df_enrichissement_final as a
        #  where exists (select *
        #                  from df_enrichissement_final as b
        #                 where b.NU_CNT_IND_CNTIS = a.NU_CNT_IND_CNTIS
        #                 and b.NU_OPE = a.NU_OPE
        #                 and b.DT_ENR_OPE = a.DT_ENR_OPE
        #                 and b.NU_ORD_GAS_OPE = a.NU_ORD_GAS_OPE
        #                )

        #         """


        # df_df_enrichissement_final = sqlContext.sql(query)

In [15]:     w = Window.partitionBy('NU_CNT_IND_CNTIS','NU_OPE','NU_ORD_GAS_OPE', 'DT_ENR_OPE')
             df_df_enrichissement_final = df_enrichissement_final.select('*', f.count('NU_CNT_IN
                 .where('dupeCount > 1')\
                 .drop('dupeCount')\
```

```
In [ ]: # df_df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_

In [ ]: # df_df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_

In [16]: df_df_enrichissement_final = df_df_enrichissement_final.where(df_df_enrichissement_fina

In [ ]: # df_df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_

In [17]: df_assemblement = df_enrichissement_uniques_rows.union(df_df_enrichissement_final)

In [ ]: # df_assemblement.count()

In [ ]: # df_assemblement.select("NU_CNT_IND_CNTIS","NU_OPE","NU_ORD_GAS_OPE","PERIO_MPA_2","DT_

In [ ]: # spark.conf.set("spark.sql.execution.arrow.enabled", "true")
        # df = df_enrichissement_final.toPandas()

In [ ]: # print(df)

In [ ]: # df.sort_values(by=['PERIO_MPA_2'], ascending=False, na_position='first')

In [ ]: # df.tail(100)

In [ ]: # df.drop_duplicates(subset =['NU_CNT_IND_CNTIS','NU_OPE','DT_ENR_OPE'],keep = 'last', i

In [ ]: # print(df)

In [ ]: # df.to_parquet('/data/dropbox/larcher/ASAC/Resultat/df_test.parquet')

In [ ]: # os.environ["SPARK_HOME"] = "/usr/local/Cellar/apache-spark/1.5.1/"
        # os.environ['PYSPARK_PYTHON'] = '/misc/anaconda3/envs/py35spark/bin/python'
        # os.environ["PYSPARK_DRIVER_PYTHON "] = '/misc/anaconda3/envs/py35spark/bin/python'

In [ ]: #spark.conf.set("spark.sql.execution.arrow.enabled", "true")
        #sqlContext = SQLContext(sc)
        # df_spark2=sqlContext.createDataFrame(pd.DataFrame(df.astype(str)))
        # df_spark2.createOrReplaceTempView('spark_df3')

In [ ]: # sqlContext.createDataFrame(pd.DataFrame(df)).show()

In [ ]: # spark.catalog.listTables()

In [ ]: # df_enrichissement_final = sqlContext.read.parquet('/data/dropbox/larcher/ASAC/Resultat

In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE",

In [ ]:

In [ ]: # df_enrichissement_final = df_enrichissement_final.orderBy("NU_CNT_IND_CNTIS","NU_OPE",

In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE",
```

```
In [ ]: #  df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE'

In [ ]: # df_enrichissement_final.printSchema()

In [ ]: # df_enrichissement_final = df_enrichissement_final.groupBy("NU_OPE","DT_ENR_OPE").max('

In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE",

In [ ]: # df_enrichissement_final = df_enrichissement_final.dropDuplicates(subset=['NU_CNT_IND_C

In [ ]: # df_enrichissement_final.count()

In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE",

In [ ]: # df_enrichissement_final.filter('NU_CNT_IND_CNTIS = 23825').count()

In [ ]: #df_enrichissement_final.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_ENR_OPE','

In [ ]: # df_inventaire_final.show(100)

In [ ]: # df_enrichissement_mpa.show()

In [ ]: # if df_enrichissement.filter("PERIO_MPA_2 is null"):
        #     df_enrichissement.drop_duplicates(subset=['NU_OPE','DT_ENR_OPE'])

In [ ]: # print(df_enrichissement.count())
        # df_enrichissement.orderBy('NU_OPE','DT_ENR_OPE', ascending=True).show(100)

In [ ]: # df_enrichissement = df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825 AND (NU_OPE = 2
        # df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_ENR_OPE','DT_DE

In [ ]: # df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_REGL_OPE','DT_D

In [ ]: # df_enrichissement = df_enrichissement.orderBy('NU_OPE','DT_REGL_OPE','PERIO_MPA_2', as

In [ ]: # df_enrichissement = df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825 AND (NU_OPE = 2
        # df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_REGL_OPE','DT_D

In [ ]: # df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825').count()

In [ ]: # df_enrichissement.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_REGL_OPE','DT_D

In [ ]: # df_enrichissement = df_enrichissement.filter('NU_CNT_IND_CNTIS = 23825')

In [ ]: # df_enrichissement.count()

In [18]: df_enrichissement_for_inventaire = df_enrichissement_for_inventaire.join(df_parquet_SAO
                                                                                 .withColumnRenamed(
                                                                                 .withColumn("FEEDER
                                                                                 .drop("NO_CONTRAT")


         df_enrichissement_for_inventaire = df_enrichissement_for_inventaire.select("DATE_SITUAT
```

```
        df_assemblement = df_assemblement.select("DATE_SITUATION","FEEDER","CD_REGIME_PROD","NU

        cols_enrichissement_for_operations = ["NO_COMPTE" ,"TYPE_COMPTE" ,"TAUX_INTERET_CALCUL"
        cols_enrichissement_for_inventaire = ["NU_OPE","NU_ORD_GAS_OPE","DT_ANNULATION_OPE","DT

        for cols in cols_enrichissement_for_operations:
            df_assemblement = df_assemblement.withColumn(cols,lit(None))

        for cols in cols_enrichissement_for_inventaire:
            df_enrichissement_for_inventaire = df_enrichissement_for_inventaire.withColumn(cols

        df_assemblement = df_assemblement.select(sorted(df_assemblement.columns))
        df_enrichissement_for_inventaire = df_enrichissement_for_inventaire.select(sorted(df_en

        #dffinal = df_enrichissement_final.unionAll(df_enrichissement_for_inventaire)
        dffinal = df_assemblement.unionAll(df_enrichissement_for_inventaire)
        dffinal = dffinal.distinct()
```

```
In [ ]: # dffinal.count()
```

```
In [ ]: # dffinal.select("NU_CNT_IND_CNTIS","NU_OPE","PERIO_MPA_2","DT_ENR_OPE","DT_DEB_EFF_MPA'
```

```
In [ ]: # dffinal.select('NU_CNT_IND_CNTIS','NU_OPE','PERIO_MPA_2','DT_REGL_OPE','DT_DEB_EFF_MPA
```

```
In [ ]: # dffinal.printSchema()
```

```
In [ ]: # df_enrichissement = df_enrichissement.select("NU_CNT_IND_CNTIS","CD_TYP_OPERATION","PE
```

```
In [ ]: # df_enrichissement_final.select("NU_CNT_IND_CNTIS","CD_TYP_OPERATION","PERIO_MPA","DT_D
```

```
In [ ]: # dffinal.distinct().count()
```

```
In [ ]: # dffinal = dffinal.filter("DT_DEB_EFF_MPA <= DT_ENR_OPE AND DT_ENR_OPE <= DT_FIN_EFF_MP
```

```
In [ ]: # dffinal = dffinal.withColumn("PERIO_MPA_2",when(dffinal["DT_ENR_OPE"].between(dffinal[
        #                   .drop("PERIO_MPA")
```

```
In [ ]: # dffinal = dffinal.filter("NU_CNT_IND_CNTIS = '23825' AND NU_OPE = 5")
```

```
In [ ]: # dffinal.count()
```

```
In [ ]: # dffinal.select("NU_CNT_IND_CNTIS","CD_TYP_OPERATION","PERIO_MPA","PERIO_MPA_2","DT_DEB
```

```
In [19]: dffinal = dffinal.select("DATE_SITUATION","FEEDER","CD_REGIME_PROD","NU_CNT_IND_CNTIS",
                          .withColumnRenamed("CD_REGIME_PROD","PRODUIT") \
                          .withColumnRenamed("NU_CNT_IND_CNTIS","NU_CONTRAT") \
                          .withColumnRenamed("DT_DEB_CNTIS","DT_DEB_CONTRAT") \
                          .withColumnRenamed("DT_FIN_CNTIS","DT_RES_CONTRAT") \
```

```
.withColumnRenamed("NU_TYP_ETAT_CNTIS","ETAT_CONTRAT") \
.withColumnRenamed("DT_SOUSC_CNTIS","DT_SOUSCRIPTION") \
.withColumnRenamed("TYP_INTERM_CNTIS","TYP_INTERMEDIAIRE") \
.withColumnRenamed("CD_INTERM_CNTIS","CD_INTERMEDIAIRE") \
.withColumnRenamed("TYP_INTERMEDIAIRE_TRANSCO","TYP_INTERMEDIAIRE_TRANSC
.withColumnRenamed("DUR_CNT_CNTIS","DUR_CNT") \
.withColumnRenamed("DT_TRM_INI_CNTIS","DT_TERM_INITIAL") \
.withColumnRenamed("DT_TRM_PRO_CNTIS","DT_TERM_PROROGE") \
.withColumnRenamed("TOP_APP_TRM_CNTIS","TOP_APPLICATION_TERME") \
.withColumnRenamed("DUREE_RENTE","DUR_RENTE") \
.withColumnRenamed("TYPE_GESTION","TYPE_GESTION") \
.withColumnRenamed("SUPPORT_INVESTISSEMENT","SUPPORT_INVESTISSEMENT") \
.withColumnRenamed("TYPE_SUPPORT","TYPE_SUPPORT") \
.withColumnRenamed("NU_GAR_GARSO","NU_GARANTIE") \
.withColumnRenamed("LBC_GAR","LIB_COURT_GARANTIE") \
.withColumnRenamed("LB_GAR","LIB_GARANTIE") \
.withColumnRenamed("TARIF","TARIF") \
.withColumnRenamed("CD_TPG_TPGGA","TUTEUR_PAQUET") \
.withColumnRenamed("CD_RSQ","RISQUE") \
.withColumnRenamed("CD_SRSQ","SOUS_RISQUE") \
.withColumnRenamed("LB_RSQ","LIB_RISQUE") \
.withColumnRenamed("TAUX_INTERET_TECHNIQUE","TX_INT_TECHNIQUE") \
.withColumnRenamed("COEFFICIENT_GARANTIE_PLANCHER","COEFF_GARANTIE_PLAN"
.withColumnRenamed("NU_PERS_PEROL","ID_ASSURE_1") \
.withColumnRenamed("NU_PERS_PEROL2","ID_ASSURE_2") \
.withColumnRenamed("DT_NAISS_PHY","DT_NAIS_ASSURE_1") \
.withColumnRenamed("DT_NAISS_PHY2","DT_NAIS_ASSURE_2") \
.withColumnRenamed("NU_TYP_ROLE_PEROL","ROLE_ASSURE_1") \
.withColumnRenamed("NU_TYP_ROLE_PEROL2","ROLE_ASSURE_2") \
.withColumnRenamed("CD_SEX_QUAL","CD_SEXE_ASSURE_1") \
.withColumnRenamed("CD_SEX_QUAL_TRANSCO","CD_SEXE_ASSURE_1_TRANSCO") \
.withColumnRenamed("CD_SEX_QUAL2","CD_SEXE_ASSURE_2") \
.withColumnRenamed("CD_SEX_QUAL_TRANSCO2","CD_SEXE_ASSURE_2_TRANSCO") \
.withColumnRenamed("NU_OPE","NU_OPERATION") \
.withColumnRenamed("DT_ANNULATION_OPE","DT_ANNULATION_OPERATION") \
.withColumnRenamed("DT_REGL_OPE","DT_REGLEMENT_OPERATION") \
.withColumnRenamed("DT_ENR_OPE","DT_ENREGISTREMENT_OPERATION") \
.withColumnRenamed("PERIO_MPA_2","PERIODICITE_PRIME") \
.withColumnRenamed("PERIO_MPA_TRANSCO","PERIODICITE_PRIME_TRANSCO") \
.withColumnRenamed("NU_TYP_OPERATI_OPE","NAT_OPERATION") \
.withColumnRenamed("CD_TYP_OPERATION","CD_TYP_OPERATION") \
.withColumnRenamed("LB_TYP_OPERATION","LIB_OPERATION") \
.withColumnRenamed("MT_OPE","MT_OPERATION") \
.withColumnRenamed("TYPE_PRIME","TYP_OPERATION") \
.withColumnRenamed("COMMISSION_GESTION","COMMISSION_GESTION") \
.withColumnRenamed("TYPE_COMMISSION_GESTION","TYP_COMMISISON_GESTION") \
.withColumnRenamed("CHARGEMENT_GESTION","CHARGEMENT_GESTION") \
.withColumnRenamed("TYPE_CHARGEMENT_GESTION","TYPE_CHARGEMENT_GESTION")
```

```
                        .withColumnRenamed("COMMISSION_PRODUIT_FINANCIER","COMMISSION_PROD_FIN")
                        .withColumnRenamed("TYPE_COMMISSION_PRODUIT_FINANCIER","TYP_COMMISSION_P
                        .withColumnRenamed("TOP_PRIMES_ARRERAGE","TOP_PRIME_ARR") \
                        .withColumnRenamed("CE_SOLDE_AU_0101","CE_PM_OUVERTURE") \
                        .withColumnRenamed("CE_SOLDE_AVANT_PRELEVEMENT","CE_PM_CLOTURE_AVANT_PRE
                        .withColumnRenamed("CE_SOLDE_APRES_PRELEVEMENT","CE_PM_CLOTURE_APRES_PRE
                        .withColumnRenamed("AV_SOLDE_AU_01_01","AV_PM_OUVERTURE") \
                        .withColumnRenamed("AV_SOLDE_RESTANT_A_REMBOURSER","AV_PM_CLOTURE")
```

```
In [ ]: # dffinal.count()
```

```
In [ ]: # dffinal.printSchema()
```

```
In [ ]: #  dffinal.distinct().filter("NU_CONTRAT = '140258'").count()
```

```
In [ ]: # dffinal = dffinal
```

```
In [ ]: # dffinal = dffinal.distinct().filter('NU_CONTRAT = 70456')
```

```
In [ ]: #dffinal.distinct().repartition(1).write.csv("/data/dropbox/larcher/ASAC/Resultat/Vue_mo
```

```
In [20]: dffinal.repartition(10).write.parquet("/data/dropbox/larcher/ASAC/Resultat/Vue_model_po
```

```
In [21]: # df = sqlContext.read.parquet("/data/dropbox/larcher/ASAC/Resultat/Vue_model_point_ASA
         # df.createOrReplaceTempView("MPASAC")
         # df.count()
```

```
Out[21]: 6340556
```

```
In [ ]: # df.filter("FEEDER = 'ASAC_SA01'").count()
```

```
In [26]: # query = """SELECT count(*)
         #              FROM MPASAC
         #              WHERE FEEDER = 'ASAC_SA03'
         #              """
         # dfresultat = sqlContext.sql(query)
```

```
In [27]: # dfresultat.show()
```

```
+--------+
|count(1)|
+--------+
|  283573|
+--------+
```

```
In [23]: dfresultat.repartition(1).write.csv("/data/dropbox/larcher/ASAC/Resultat/Vue_model_poin
```

```
In [3]: df_vue = sqlContext.read.parquet("/data/dropbox/larcher/ASAC/Resultat/Vue_model_point_AS
```

```
In [4]: df_vue.printSchema()
```

```
root
 |-- DATE_SITUATION: date (nullable = true)
 |-- FEEDER: string (nullable = true)
 |-- PRODUIT: string (nullable = true)
 |-- NU_CONTRAT: integer (nullable = true)
 |-- DT_DEB_CONTRAT: date (nullable = true)
 |-- DT_RES_CONTRAT: date (nullable = true)
 |-- ETAT_CONTRAT: integer (nullable = true)
 |-- DT_SOUSCRIPTION: date (nullable = true)
 |-- TYP_INTERMEDIAIRE: integer (nullable = true)
 |-- CD_INTERMEDIAIRE: string (nullable = true)
 |-- TYP_INTERMEDIAIRE_TRANSCO: string (nullable = true)
 |-- DUR_CNT: integer (nullable = true)
 |-- DT_TERM_INITIAL: date (nullable = true)
 |-- DT_TERM_PROROGE: date (nullable = true)
 |-- TOP_APPLICATION_TERME: string (nullable = true)
 |-- DUR_RENTE: string (nullable = true)
 |-- TYPE_GESTION: string (nullable = true)
 |-- SUPPORT_INVESTISSEMENT: string (nullable = true)
 |-- TYPE_SUPPORT: string (nullable = true)
 |-- NU_GARANTIE: integer (nullable = true)
 |-- LIB_COURT_GARANTIE: string (nullable = true)
 |-- LIB_GARANTIE: string (nullable = true)
 |-- TARIF: string (nullable = true)
 |-- TUTEUR_PAQUET: string (nullable = true)
 |-- RISQUE: string (nullable = true)
 |-- SOUS_RISQUE: string (nullable = true)
 |-- LIB_RISQUE: string (nullable = true)
 |-- TX_INT_TECHNIQUE: string (nullable = true)
 |-- COEFF_GARANTIE_PLAN: string (nullable = true)
 |-- ID_ASSURE_1: decimal(10,0) (nullable = true)
 |-- ID_ASSURE_2: string (nullable = true)
 |-- DT_NAIS_ASSURE_1: date (nullable = true)
 |-- DT_NAIS_ASSURE_2: string (nullable = true)
 |-- ROLE_ASSURE_1: integer (nullable = true)
 |-- ROLE_ASSURE_2: string (nullable = true)
 |-- CD_SEXE_ASSURE_1: string (nullable = true)
 |-- CD_SEXE_ASSURE_1_TRANSCO: integer (nullable = true)
 |-- CD_SEXE_ASSURE_2: string (nullable = true)
 |-- CD_SEXE_ASSURE_2_TRANSCO: string (nullable = true)
 |-- NU_OPERATION: integer (nullable = true)
 |-- DT_ANNULATION_OPERATION: date (nullable = true)
 |-- DT_REGLEMENT_OPERATION: date (nullable = true)
 |-- DT_ENREGISTREMENT_OPERATION: date (nullable = true)
 |-- PERIODICITE_PRIME: string (nullable = true)
 |-- PERIODICITE_PRIME_TRANSCO: string (nullable = true)
```

```
|-- NAT_OPERATION: integer (nullable = true)
|-- CD_TYP_OPERATION: string (nullable = true)
|-- LIB_OPERATION: string (nullable = true)
|-- MT_OPERATION: decimal(11,3) (nullable = true)
|-- TYP_OPERATION: string (nullable = true)
|-- COMMISSION_GESTION: string (nullable = true)
|-- TYP_COMMISISON_GESTION: string (nullable = true)
|-- CHARGEMENT_GESTION: string (nullable = true)
|-- TYPE_CHARGEMENT_GESTION: string (nullable = true)
|-- COMMISSION_PROD_FIN: string (nullable = true)
|-- TYP_COMMISSION_PROD_FIN: string (nullable = true)
|-- TOP_PRIME_ARR: string (nullable = true)
|-- NO_COMPTE: string (nullable = true)
|-- TYPE_COMPTE: string (nullable = true)
|-- TAUX_INTERET_CALCUL: decimal(8,5) (nullable = true)
|-- TAUX_BRUT: decimal(8,5) (nullable = true)
|-- TAUX_GESTION_FINANCIERE: decimal(8,5) (nullable = true)
|-- CE_PM_OUVERTURE: decimal(15,2) (nullable = true)
|-- CE_VERSEMENT_BRUT: decimal(15,2) (nullable = true)
|-- CE_FRAIS_PROMOTION: decimal(15,2) (nullable = true)
|-- CE_FRAIS_GESTION: decimal(15,2) (nullable = true)
|-- CE_FRAIS_COMMISSION: decimal(15,2) (nullable = true)
|-- CE_VERSEMENT_NET: decimal(15,2) (nullable = true)
|-- CE_COT_STAT_PRELEVEE_SUR_VERSEMENT: decimal(15,2) (nullable = true)
|-- CE_ABON_REVUE_PRELEVE_SUR_VERSEMENT: decimal(15,2) (nullable = true)
|-- CE_TRANSFERT: decimal(15,2) (nullable = true)
|-- CE_RETRAIT_TOTAL: decimal(15,2) (nullable = true)
|-- CE_RETRAIT_TOTAL_TERME: decimal(15,2) (nullable = true)
|-- CE_RETRAIT_TOTAL_LOI_FINANCE: decimal(15,2) (nullable = true)
|-- CE_RETRAIT_PARTIEL: decimal(15,2) (nullable = true)
|-- CE_PFL: decimal(15,2) (nullable = true)
|-- CE_PFU_REGLE: decimal(15,2) (nullable = true)
|-- CE_PFU_DISPENSE: decimal(15,2) (nullable = true)
|-- CE_VIREMENT_RESERVE: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_BENEF: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_PS: decimal(15,2) (nullable = true)
|-- CE_SINSITRE_CA_FISCALITE: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_TCDC: decimal(15,2) (nullable = true)
|-- CE_INTERETS_DISTRIBUES: decimal(15,2) (nullable = true)
|-- CE_PM_CLOTURE_AVANT_PRELEVEMENT: decimal(15,2) (nullable = true)
|-- CE_MONTANT_CRDS: decimal(15,2) (nullable = true)
|-- CE_TAUX_CRDS: decimal(8,5) (nullable = true)
|-- CE_MONTANT_CSG: decimal(15,2) (nullable = true)
|-- CE_TAUX_CSG: decimal(8,5) (nullable = true)
|-- CE_MONTANT_PS: decimal(15,2) (nullable = true)
|-- CE_TAUX_PS: decimal(8,5) (nullable = true)
|-- CE_MONTANT_PSOL: decimal(15,2) (nullable = true)
|-- CE_TAUX_PSOL: decimal(8,5) (nullable = true)
```

```
 |-- CE_COT_STAT_PRELEVEE_SUR_INTERETS: decimal(15,2) (nullable = true)
 |-- CE_ABON_REVUE_PRELEVE_SUR_INTERETS: decimal(15,2) (nullable = true)
 |-- CE_PM_CLOTURE_APRES_PRELEVEMENT: decimal(15,2) (nullable = true)
 |-- AV_PM_OUVERTURE: decimal(15,2) (nullable = true)
 |-- AV_AVANCE_CONSENTIE: decimal(15,2) (nullable = true)
 |-- AV_REMB_CAPITAL: decimal(15,2) (nullable = true)
 |-- AV_REMB_INTERETS: decimal(15,2) (nullable = true)
 |-- AV_INTERETS_DUS: decimal(15,2) (nullable = true)
 |-- AV_PM_CLOTURE: decimal(15,2) (nullable = true)
```

```
In [4]: df = spark.read.parquet("/data/prod_env/data/edited_data/data_prep/SA03/SA03_ASAC_flux_i
```

```
In [5]: df.printSchema()
```

```
root
 |-- DATE_DEBUT_ARRETE: date (nullable = true)
 |-- DATE_FIN_ARRETE: date (nullable = true)
 |-- DATE_PASSAGE: date (nullable = true)
 |-- NO_PERSONNE: string (nullable = true)
 |-- NO_CONTRAT: string (nullable = true)
 |-- CODE_CONTRAT: string (nullable = true)
 |-- DATE_DEBUT_EFFET_CONTRAT: date (nullable = true)
 |-- DATE_FIN_CONTRAT: date (nullable = true)
 |-- MOTIF_FIN_CONTRAT: string (nullable = true)
 |-- NO_PRODUIT: string (nullable = true)
 |-- NO_CONVENTION: string (nullable = true)
 |-- NO_COMPTE: string (nullable = true)
 |-- TYPE_COMPTE: string (nullable = true)
 |-- TUTEUR_PAQUET: string (nullable = true)
 |-- NO_GARANTIE: string (nullable = true)
 |-- TAUX_INTERET_CALCUL: decimal(8,5) (nullable = true)
 |-- TAUX_BRUT: decimal(8,5) (nullable = true)
 |-- TAUX_GESTION_FINANCIERE: decimal(8,5) (nullable = true)
 |-- CE_SOLDE_AU_0101: decimal(15,2) (nullable = true)
 |-- CE_VERSEMENT_BRUT: decimal(15,2) (nullable = true)
 |-- CE_FRAIS_PROMOTION: decimal(15,2) (nullable = true)
 |-- CE_FRAIS_GESTION: decimal(15,2) (nullable = true)
 |-- CE_FRAIS_COMMISSION: decimal(15,2) (nullable = true)
 |-- CE_VERSEMENT_NET: decimal(15,2) (nullable = true)
 |-- CE_COT_STAT_PRELEVEE_SUR_VERSEMENT: decimal(15,2) (nullable = true)
 |-- CE_ABON_REVUE_PRELEVE_SUR_VERSEMENT: decimal(15,2) (nullable = true)
 |-- CE_TRANSFERT: decimal(15,2) (nullable = true)
 |-- CE_RETRAIT_TOTAL: decimal(15,2) (nullable = true)
 |-- CE_RETRAIT_TOTAL_TERME: decimal(15,2) (nullable = true)
 |-- CE_RETRAIT_TOTAL_LOI_FINANCE: decimal(15,2) (nullable = true)
 |-- CE_RETRAIT_PARTIEL: decimal(15,2) (nullable = true)
```

```
|-- CE_PFL: decimal(15,2) (nullable = true)
|-- CE_PFU_REGLE: decimal(15,2) (nullable = true)
|-- CE_PFU_DISPENSE: decimal(15,2) (nullable = true)
|-- CE_VIREMENT_RESERVE: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_BENEF: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_PS: decimal(15,2) (nullable = true)
|-- CE_SINSITRE_CA_FISCALITE: decimal(15,2) (nullable = true)
|-- CE_SINISTRE_CA_TCDC: decimal(15,2) (nullable = true)
|-- CE_INTERETS_DISTRIBUES: decimal(15,2) (nullable = true)
|-- CE_SOLDE_AVANT_PRELEVEMENT: decimal(15,2) (nullable = true)
|-- CE_MONTANT_CRDS: decimal(15,2) (nullable = true)
|-- CE_TAUX_CRDS: decimal(8,5) (nullable = true)
|-- CE_MONTANT_CSG: decimal(15,2) (nullable = true)
|-- CE_TAUX_CSG: decimal(8,5) (nullable = true)
|-- CE_MONTANT_PS: decimal(15,2) (nullable = true)
|-- CE_TAUX_PS: decimal(8,5) (nullable = true)
|-- CE_MONTANT_PSOL: decimal(15,2) (nullable = true)
|-- CE_TAUX_PSOL: decimal(8,5) (nullable = true)
|-- CE_COT_STAT_PRELEVEE_SUR_INTERETS: decimal(15,2) (nullable = true)
|-- CE_ABON_REVUE_PRELEVE_SUR_INTERETS: decimal(15,2) (nullable = true)
|-- CE_SOLDE_APRES_PRELEVEMENT: decimal(15,2) (nullable = true)
|-- AV_SOLDE_AU_01_01: decimal(15,2) (nullable = true)
|-- AV_AVANCE_CONSENTIE: decimal(15,2) (nullable = true)
|-- AV_REMB_CAPITAL: decimal(15,2) (nullable = true)
|-- AV_REMB_INTERETS: decimal(15,2) (nullable = true)
|-- AV_INTERETS_DUS: decimal(15,2) (nullable = true)
|-- AV_SOLDE_RESTANT_A_REMBOURSER: decimal(15,2) (nullable = true)
|-- DATE_DEBUT_ARRETE_AUDIT: string (nullable = true)
|-- DATE_FIN_ARRETE_AUDIT: string (nullable = true)
|-- DATE_PASSAGE_AUDIT: string (nullable = true)
|-- DATE_DEBUT_EFFET_CONTRAT_AUDIT: string (nullable = true)
|-- DATE_FIN_CONTRAT_AUDIT: string (nullable = true)
|-- TAUX_INTERET_CALCUL_AUDIT: string (nullable = true)
|-- TAUX_BRUT_AUDIT: string (nullable = true)
|-- TAUX_GESTION_FINANCIERE_AUDIT: string (nullable = true)
|-- CE_SOLDE_AU_0101_AUDIT: string (nullable = true)
|-- CE_VERSEMENT_BRUT_AUDIT: string (nullable = true)
|-- CE_FRAIS_PROMOTION_AUDIT: string (nullable = true)
|-- CE_FRAIS_GESTION_AUDIT: string (nullable = true)
|-- CE_FRAIS_COMMISSION_AUDIT: string (nullable = true)
|-- CE_VERSEMENT_NET_AUDIT: string (nullable = true)
|-- CE_COT_STAT_PRELEVEE_SUR_VERSEMENT_AUDIT: string (nullable = true)
|-- CE_ABON_REVUE_PRELEVE_SUR_VERSEMENT_AUDIT: string (nullable = true)
|-- CE_TRANSFERT_AUDIT: string (nullable = true)
|-- CE_RETRAIT_TOTAL_AUDIT: string (nullable = true)
|-- CE_RETRAIT_TOTAL_TERME_AUDIT: string (nullable = true)
|-- CE_RETRAIT_TOTAL_LOI_FINANCE_AUDIT: string (nullable = true)
|-- CE_RETRAIT_PARTIEL_AUDIT: string (nullable = true)
```

```
 |-- CE_PFL_AUDIT: string (nullable = true)
 |-- CE_PFU_REGLE_AUDIT: string (nullable = true)
 |-- CE_PFU_DISPENSE_AUDIT: string (nullable = true)
 |-- CE_VIREMENT_RESERVE_AUDIT: string (nullable = true)
 |-- CE_SINISTRE_CA_BENEF_AUDIT: string (nullable = true)
 |-- CE_SINISTRE_CA_PS_AUDIT: string (nullable = true)
 |-- CE_SINSITRE_CA_FISCALITE_AUDIT: string (nullable = true)
 |-- CE_SINISTRE_CA_TCDC_AUDIT: string (nullable = true)
 |-- CE_INTERETS_DISTRIBUES_AUDIT: string (nullable = true)
 |-- CE_SOLDE_AVANT_PRELEVEMENT_AUDIT: string (nullable = true)
 |-- CE_MONTANT_CRDS_AUDIT: string (nullable = true)
 |-- CE_TAUX_CRDS_AUDIT: string (nullable = true)
 |-- CE_MONTANT_CSG_AUDIT: string (nullable = true)
 |-- CE_TAUX_CSG_AUDIT: string (nullable = true)
 |-- CE_MONTANT_PS_AUDIT: string (nullable = true)
 |-- CE_TAUX_PS_AUDIT: string (nullable = true)
 |-- CE_MONTANT_PSOL_AUDIT: string (nullable = true)
 |-- CE_TAUX_PSOL_AUDIT: string (nullable = true)
 |-- CE_COT_STAT_PRELEVEE_SUR_INTERETS_AUDIT: string (nullable = true)
 |-- CE_ABON_REVUE_PRELEVE_SUR_INTERETS_AUDIT: string (nullable = true)
 |-- CE_SOLDE_APRES_PRELEVEMENT_AUDIT: string (nullable = true)
 |-- AV_SOLDE_AU_01_01_AUDIT: string (nullable = true)
 |-- AV_AVANCE_CONSENTIE_AUDIT: string (nullable = true)
 |-- AV_REMB_CAPITAL_AUDIT: string (nullable = true)
 |-- AV_REMB_INTERETS_AUDIT: string (nullable = true)
 |-- AV_INTERETS_DUS_AUDIT: string (nullable = true)
 |-- AV_SOLDE_RESTANT_A_REMBOURSER_AUDIT: string (nullable = true)
 |-- AUDIT_SRC_FILE: string (nullable = true)
 |-- MODULE_START: string (nullable = true)
 |-- INGESTION_START: string (nullable = true)
 |-- DATE_SITUATION: date (nullable = true)
 |-- EXER: string (nullable = true)
```

```
In [7]: df.select("DATE_SITUATION","EXER").distinct().orderBy("EXER").show()
```

```
+--------------+----+
|DATE_SITUATION|EXER|
+--------------+----+
|    2019-12-29|2018|
|    2019-12-29|2019|
|    2020-09-30|2020|
|    2020-01-31|2020|
|    2020-06-30|2020|
|    2020-02-29|2020|
|    2020-05-31|2020|
|    2020-08-31|2020|
```

```
|    2020-03-31|2020|
|    2020-04-30|2020|
|    2020-07-31|2020|
+--------------+----+
```

```
In [8]: spark.stop()
```