In [24]:
```
# sc.list_packages()
sc.install_pypi_package("pandas==0.25.1")
sc.install_pypi_package("boto3==1.26.7")
sc.install_pypi_package("rapidfuzz")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

```
An error was encountered:
Package already installed for current Spark context!
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/context.py", line 111
0, in install_pypi_package
    raise ValueError("Package already installed for current Spark contex
t!")
ValueError: Package already installed for current Spark context!
```

In [25]:
```
confs = ["spark.yarn.executor.memoryOverhead","spark.sql.shuffle.partitions"
print(sc._conf.get('spark.app.name'))
print(sc._conf.get('spark.submit.deployMode'))
print(sc._conf.get("spark.yarn.executor.memoryOverhead"))
print(sc._conf.get("spark.sql.files.maxPartitionBytes"))
for conf in confs:
    print(conf,":",sc._conf.get(conf))
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
ProcessingData
client
None
None
spark.yarn.executor.memoryOverhead : None
spark.sql.shuffle.partitions : None
spark.network.timeout : 100800s
spark.driver.memory : 122000M
```

Imports

In [26]:
```
from pyspark import SparkContext
from pyspark.sql import SQLContext
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import year, month, dayofmonth
import boto3
import re
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, ArrayType, MapType, IntegerType, D
from pyspark.sql.types import *
from pyspark.sql import functions as F
from collections import Counter
import pyspark
import rapidfuzz
from rapidfuzz import fuzz
import time
import random

# import s3fs
# import boto3
#How to pip install while creation of cluster
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [27]:
```
#         .config("spark.memory.offHeap.enabled","true") \
#         .config("spark.memory.offHeap.size","10g") \
# .config("spark.submit.deployMode","client") \
#        .config("spark.driver.memory","200g") \
#        .config("spark.executor.memory","50g") \
#        .config("spark.emr.default.executor.memory", '50g') \

# spark.conf.set("spark.sql.shuffle.partitions",100)

#Number of partitions should be equal to or greater than the cores to achiev
# config = pyspark.SparkConf().setAll([('spark.executor.memory', '80'), ('sp
#                                       ('spark.cores.max', '240'), ('spark.c
# spark.sparkContext.stop()
config = pyspark.SparkConf().setAll([('spark.network.timeout', '100800s'), (
spark = SparkSession.builder.config(conf=config).appName('ProcessingData') \
        .getOrCreate()
# spark = SparkSession.builder.getOrCreate()
# sc = pyspark.SparkContext.getOrCreate(conf=config)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

Task

Read data of specific countries. Within them, if any org in any row is a close match to the list of primary or secondary firms then return a one. Groupby with average tone, and count for total, primary, and secondary firms. Groupby level, country, month. Part2: Same as previous but with Indian, Indians, Foreign, Pharmaceutical themes

Countries:

1. Gambia - GA
2. South Africa - SF
3. Cameroon - CM
4. Senegal - SG
5. Malawi - MI
6. Mozambique - MZ

1. Reading Files

In [28]:
```
from boto3 import client

def return_yearly_parquet_files(year):
    print(f"Return file list for the year: {year}")
    file_list = []
    conn = client('s3')  # again assumes boto.cfg setup, assume AWS S3
    for key in conn.list_objects(Bucket='kcsra', Prefix = f"{str(year)} GDEL
        file = "s3://kcsra/"+ key['Key']
        if "parquet" in file:
            file_list.append(file)
    return file_list

def return_correct_file_paths(year_file_list):
    print("Returning the working parquet files ...")
    correct_file_paths = []
    for i,file in enumerate(year_file_list):

        try:
            df = spark.read.parquet(file)
            correct_file_paths.append(file)
        except Exception as e:
            print(f"[{str(i+1)}/{len(year_file_list)}]")
            print("->ERROR while reading:")
            print(file)
            print()
    return correct_file_paths

def return_df(correct_file_paths, req_cols, grid_id = None ):
    print("Constructing a dataframe from required file paths")

    df = spark.read.parquet(*correct_file_paths).select(*req_cols).dropDupli

    if grid_id:
        print("Filter Grid ID:", grid_id)
        df = df.filter(df.GRID_IDs == grid_id)

    df = df.withColumn('day',dayofmonth(df.date)).withColumn('month',month(c
    print("Total rows:",df.count())
    # df.printSchema()
    return df

def extract_tone(tone_string):
    try:
        tone_float = float(re.findall("tone=(.*?),",tone_string)[0])
        return tone_float
    except Exception as e:
        print("Error extracting tone:" + str(exception))
        print(tone_string)
        print("="*20)
        return None

def save_tone_in_new_col(df):
    print("Saving the tone in a new column")
    extract_tone_udf = udf(extract_tone, DoubleType())
    df = df.withColumn('Extracted_tone', extract_tone_udf(df['tone'])).drop(
    return df

@udf(returnType=ArrayType(StringType()))
def retrieve_theme_list(themes):
    if not themes:
        return {}
    themes = str(themes)
    return list(Counter(re.findall(f"theme=(.*?),",themes)).keys())

@udf(returnType=ArrayType(StringType()))
def retrieve_organization_list(organizations):
    if not organizations:
        return {}
    organizations = str(organizations)
    return list(Counter(re.findall(f"organization=(.*?),",organizations)).ke

@udf(returnType=StringType())
def aggregate_category(category):
    super_counter = Counter({})
    for row in category:
        super_counter += Counter(row)
    return ", ".join([f"{categ[0]}:{categ[1]}" for categ in super_counter.mo


def are_strings_similar(target_str, data_str, match_threshold):
    target_str, data_str = str(target_str).lower(), str(data_str).lower()
    partial_match, sorted_match = fuzz.partial_ratio(target_str, data_str),

    if max(partial_match, sorted_match) >= match_threshold:
        # print(max(partial_match, sorted_match))
        return True
    else:
        return False


def compare_string_matches_lists(target_list, data_list, match_threshold = 7
    for target_str in target_list:
        for data_str in data_list:
            if are_strings_similar(target_str, data_str, match_threshold):
                # print("Match Found", f"| {target_str} : {data_str}")
                return 1
    return 0


def udf_compare_strings_lists(target_list, match_threshold):
    return udf(lambda data_list: compare_string_matches_lists(target_list, d
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [29]:
```
target_str = "Promac"
data_str = "promac llc"
```

```
match_threshold = 78
are_strings_similar(target_str, data_str, match_threshold)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
True
```

MAIN

In [30]:
```
#Collect all the orgs in a list and run the matcher function to label primar
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [31]:
```
primary_firms = """Cadila
Cipla
Dr. Reddy's
Ipca Lab
Lupin
pharma dynamics
Maiden
Orchid
Panacea Biotech
Ranbaxy
Sun Pharma"""

secondary_firms = """Actiza
Ajanta Pharma
Atul
Aurobindo
Bal Pharma
Glenmark
Global Pharma
InterMed Pharma
JoinHub Pharma
KEC Ltd.
Kirloskar Bros
Maan Pharma
MS Pharma
Promac
Pyramid Pharma
SenegIndia
XL Laboratories
Weefsel Pharma"""

indian_themes = """Indian"""

indian_foreign_themes = """foreign invest"""

indian_pharmaceuticals_themes = """pharmaceuticals"""

prim_firm_list = [firm.strip() for firm in primary_firms.split("\n")]
sec_firm_list = [firm.strip() for firm in secondary_firms.split("\n")]
```

```
indian_theme_list = [theme.strip() for theme in indian_themes.split("\n")]
indian_foreign_theme_list = [theme.strip() for theme in indian_foreign_theme
indian_pharmaceuticals_theme_list = [theme.strip() for theme in indian_pharm
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [32]:

```
# match_str = """[Google, Ballistic Research Centre, Institute Of Forensic S

# data_list = [x.strip() for x in match_str.strip().strip("[").strip("]").sp
# compare_string_matches_lists(prim_firm_list, data_list, match_threshold =
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [34]:

```
countries = [ "SF"] #["CM","GA","MZ","MI","SG"]
match_threshold = 91
req_cols = ['gkgrecordid','date','tone','country', 'year','GRID_IDs', 'organ
t0 = time.time()
for country in countries:
    for year in range(2021, 2023):
        t1 = time.time()
        print(f"Country: {country} | Year: {year}")
        file_list = return_yearly_parquet_files(year)
        file_list2 = [file for file in file_list if country in file]
        correct_file_paths = return_correct_file_paths(file_list2)
        correct_file_paths = random.sample(correct_file_paths, int(len(corre
        df = return_df(correct_file_paths, req_cols)
        df = save_tone_in_new_col(df)
        df = df.dropna(subset = ["tone"])
        req_cols_grpby = ['country', 'day', 'month', 'year', 'tone', 'Primar
                          'IndianThemes','IndianForeignThemes','IndianPharma
        t2 = time.time()

        print("Time to read files:", t2-t1)
        df2 = df.withColumn("Org_list", retrieve_organization_list(F.col("or
                .withColumn("Theme_list", retrieve_theme_list(F.col("themes"
        # df.unpersist()
        df3 = df2.withColumn('PrimaryFirms', udf_compare_strings_lists(prim_
            .withColumn('SecondaryFirms', udf_compare_strings_lists(sec_firm
            .withColumn('IndianThemes', udf_compare_strings_lists(indian_the
            .withColumn('IndianForeignThemes', F.col("IndianThemes") * udf_c
            .withColumn('IndianPharmaThemes', F.col("IndianThemes") * udf_co
            .select(*req_cols_grpby)
        # df2.unpersist()

        df4 = df3.withColumn("PrimaryFirmTone", F.col("PrimaryFirms") * F.co
                 .withColumn("SecondaryFirmTone", F.col("SecondaryFirms") *
                 .withColumn("IndianThemesTone", F.col("IndianThemes") * F.c
                 .withColumn("IndianForeignThemesTone", F.col("IndianForeign
                 .withColumn("IndianPharmaThemesTone", F.col("IndianPharmaTh

        t3 = time.time()
        print("T3 time:", t3-t2)
        # df3.unpersist()
        aggregation_cols = ['country','month','year']
        df5 = df4.groupby(*aggregation_cols).agg(F.count(F.col("tone")).alia
                                  F.avg('tone').alias('AverageSentiment'),\
                                  F.sum('PrimaryFirms').alias('PrimaryFirms_
                                   F.sum('PrimaryFirmTone').alias('SumPrimar
                                  F.sum('SecondaryFirms').alias('SecondaryFi
                                   F.sum('SecondaryFirmTone').alias('SumSecc
                                  F.sum('IndianThemes').alias('IndianThemes_
                                   F.sum('IndianThemesTone').alias('SumIndia
                                  F.sum('IndianForeignThemes').alias('Indiar
                                   F.sum('IndianForeignThemesTone').alias('S
                                  F.sum('IndianPharmaThemes').alias('IndianF
                                   F.sum('IndianPharmaThemesTone').alias('Su
        # df4.unpersist()
        df6 = df5.withColumn("AverageSentiment_PrimaryFirms", F.col("SumPrim
            .withColumn("AverageSentiment_SecondaryFirms", F.col("SumSeconda
            .withColumn("AverageSentiment_IndianThemes", F.col("SumIndianThe
            .withColumn("AverageSentiment_IndianForeignThemes", F.col("SumIr
            .withColumn("AverageSentiment_IndianPharmaThemes", F.col("SumInc
            .drop("SumPrimaryFirmTone","SumSecondaryFirmTone","SumIndianThem
        # df5.unpersist()
        t4 = time.time()
        print("T4 time:", t4-t3)

        df6.repartition(1).write.option("header",True) \
           .csv(f"s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_Themes_
        df6.unpersist()
        t5 = time.time()
        print("T5 time:", t5-t4)
        print("="*50)

        # break
print("Total time", time.time()-t0)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

```
An error was encountered:
'Unable to infer schema for Parquet. It must be specified manually.;'
Traceback (most recent call last):
  File "<stdin>", line 31, in return_df
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/readwriter.py", l
ine 316, in parquet
    return self._df(self._jreader.parquet(_to_seq(self._spark._sc, paths)))
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.p
y", line 1257, in __call__
    answer, self.gateway_client, self.target_id, self.name)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/utils.py", line 6
9, in deco
    raise AnalysisException(s.split(': ', 1)[1], stackTrace)
pyspark.sql.utils.AnalysisException: 'Unable to infer schema for Parquet. I
t must be specified manually.;'
```

In [13]:
```
cols = ['PrimaryFirms', 'SecondaryFirms', 'IndianThemes', 'IndianForeignThem
agg_cols = {col:'sum' for col in cols}
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

In [12]:
```
df3.agg(agg_cols).show()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
+-------------------------+--------------------+------------------------+-----
------------+-----------------+
|sum(IndianForeignThemes)|sum(SecondaryFirms)|sum(IndianPharmaThemes)|sum(I
ndianThemes)|sum(PrimaryFirms)|
+-------------------------+--------------------+------------------------+-----
------------+-----------------+
|                      98|                 203|                    1794|
5899|              193|
+-------------------------+--------------------+------------------------+-----
------------+-----------------+
```

Joining Files

Country Level

In [21]:
```
countries = ["MZ", "MI", "SG","GA","SF","CM"]
for country in countries:
    df_combined = spark.read.option("header",True) \
                .csv(f"s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_The
    df_combined.repartition(1).write.option("header",True) \
                .csv(f"s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_The
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

```
An error was encountered:
'path s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_Themes_Sentiment/M
Z/Monthly/AllYears already exists.;'
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/readwriter.py", l
ine 935, in csv
    self._jwrite.csv(path)
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.p
y", line 1257, in __call__
    answer, self.gateway_client, self.target_id, self.name)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/utils.py", line 6
9, in deco
    raise AnalysisException(s.split(': ', 1)[1], stackTrace)
pyspark.sql.utils.AnalysisException: 'path s3://shrivats-dev/Tasks/23_Prima
ry_Secondary_Firms_Themes_Sentiment/MZ/Monthly/AllYears already exists.;'
```

All

```
In [35]: df_combined_all_countries = spark.read.option("header",True) \
                        .csv(f"s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_The
         df_combined_all_countries = df_combined_all_countries.withColumn("month", F.
         df_combined_all_countries_sorted = df_combined_all_countries.sort("country",
         df_combined_all_countries_sorted.repartition(1).write.option("header",True)
                        .csv(f"s3://shrivats-dev/Tasks/23_Primary_Secondary_Firms_The
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=
Layout(height='25px', width='50%'),…
```

```
In [ ]:
```