

## P3-D2

December 12, 2017

## 1 GDELT Event Data Analysis

```
In [1]: from pyspark.sql.types import StructType, StructField, FloatType, LongType, StringType
        from pyspark.shell import spark
```

Welcome to

```

      _ _ _ _ _
     /  _  /  _ _ _ _ _ _ _ _ _ /  /  _ _
    _ \  \  _ \  _ \  _ \  _ \  _ \  _ \  _ \
   /  _  /  _ \  _ \  _ \  _ \  _ \  _ \  _ \
      /  _  /  _ \  _ \  _ \  _ \  _ \  _ \
      /  _  /  _ \  _ \  _ \  _ \  _ \  _ \
                                     version 2.2.0

```

```
Using Python version 3.6.3 (default, Oct 6 2017 12:04:38)
SparkSession available as 'spark'.
```

```
In [2]: attrs = []
        f = open('./GDELT-EVENTS-ATTRIBUTES.txt')

        for line in f:
            tokens = line.split(',')
            if tokens[1].strip() == "INTEGER":
                attrs.append(StructField(tokens[0].strip(), IntegerType(), True))
            elif tokens[1].strip() == "STRING":
                attrs.append(StructField(tokens[0].strip(), StringType(), True))
            elif tokens[1].strip() == "FLOAT":
                attrs.append(StructField(tokens[0].strip(), FloatType(), True))

        schema = StructType(attrs)
        schema
```

```
Out[2]: StructType(List(StructField(GLOBALEVENTID,IntegerType,true),StructField(SQLDATE,IntegerType,true)))
```

```
In [3]: feats = []
        feats.append(StructField('Country_Code', StringType(), True))
        feats.append(StructField('Country_Name', StringType(), True))
        schema_df_codes = StructType(feats)
```

```

In [4]: df_country_codes = spark.read.format('CSV').option('sep', ',').schema(schema_df_codes)

In [6]: data_aids = [('070', 'Provide Aid'),('071','Provide Economic Aid'),
                    ('072','Provide Military Aid'),('073','Provide humanitarian aid'),
                    ('074','Provide military protection or peacekeeping'),
                    ('075','Grant Asylum')]
df_aids = spark.createDataFrame(data_aids,['Event_Code','Type_of_Event'])

In [7]: data_deport = [('174','Expel or deport individuals')]
df_deport = spark.createDataFrame(data_deport,['Event_Code','Type_of_Event'])

In [8]: data_destrn = [('200','Use massive unconventional force'),
                      ('201','Engage in mass expulsion'),
                      ('202','Engage in mass killings'),
                      ('203','Engage in ethnic cleansing'),
                      ('204','Use weapons of mass destruction')]
df_massdestrn = spark.createDataFrame(data_destrn,['Event_Code','Type_of_Event'])

In [10]: from pyspark.sql import Column as col
from pyspark.sql.functions import udf
import pyspark.sql.functions as fun

def concat_asc_codes(str1,str2):
    list_codes = []
    list_codes.append(str1)
    list_codes.append(str2)
    list_codes.sort()
    return list_codes[0]+"_"+list_codes[1]

def do_aid_analysis(df,event_code,map_names_event):
    #Filter events required for analysis
    #Group using country codes and event base code
    #joining to country code table to get names of country codes
    aid_maps_1 = df.select('*').where(df.EventRootCode == event_code).groupBy(df.Actor1Geo_CountryCode,
                                     df.Actor2Geo_CountryCode,
                                     df.EventBaseCode).agg(fun.count('*').alias('no_of_events'),
                                     fun.sum('GoldsteinScale').alias('TotalGoldsteinScale'),
                                     fun.sum('NumMentions').alias('TotalNumMentions'),
                                     fun.sum('NumSources').alias('TotalNumSources'),
                                     fun.sum('NumArticles').alias('TotalNumArticles'),
                                     fun.sum('AvgTone').alias('TotalAvgTone'))
    aid_maps_1 = aid_maps_1.select(aid_maps_1.Actor1Geo_CountryCode ,
                                   aid_maps_1.Country_Name.alias("Actor1Geo_Country_Name"),
                                   aid_maps_1.Actor2Geo_CountryCode ,
                                   aid_maps_1.EventBaseCode ,
                                   aid_maps_1.no_of_events,
                                   aid_maps_1.TotalGoldsteinScale,
                                   aid_maps_1.TotalNumMentions,

```

```

        aid_maps_1.TotalNumSources,
        aid_maps_1.TotalNumArticles,
        aid_maps_1.TotalAvgTone)
aid_maps_1 = aid_maps_1.join(df_country_codes,aid_maps_1.Actor2Geo_CountryCode ==
aid_maps = aid_maps_1.select(
    aid_maps_1.Actor1Geo_CountryCode ,
    aid_maps_1.Actor1Geo_Country_Name,
    aid_maps_1.Actor2Geo_CountryCode ,
    aid_maps_1.Country_Name.alias("Actor2Geo_Country_Name"),
    aid_maps_1.EventBaseCode ,
    aid_maps_1.no_of_events,
    aid_maps_1.TotalGoldsteinScale,
    aid_maps_1.TotalNumMentions,
    aid_maps_1.TotalNumSources,
    aid_maps_1.TotalNumArticles,
    aid_maps_1.TotalAvgTone)
#Sorted based on number of events
sorted_aid_maps = aid_maps.sort(aid_maps.no_of_events.desc())
#Filtered relations within same country
diff_actr1_actr2 = sorted_aid_maps.filter(sorted_aid_maps.Actor1Geo_CountryCode !=
diff_actr1_actr2_copy = diff_actr1_actr2
diff_actr1_actr2_copy = diff_actr1_actr2_copy.select(diff_actr1_actr2_copy.Actor1Geo_CountryCode,
    diff_actr1_actr2_copy.Actor1Geo_Country_Name.alias("CountryNameA1"),
    diff_actr1_actr2_copy.Actor2Geo_CountryCode.alias("CountryCodeA2"),
    diff_actr1_actr2_copy.Actor2Geo_Country_Name.alias("CountryNameA2"),
    diff_actr1_actr2_copy.EventBaseCode.alias("Event_Base_Code"),
    diff_actr1_actr2_copy.TotalGoldsteinScale.alias("total_goldsteinscale"),
    diff_actr1_actr2_copy.no_of_events.alias("total_number_of_events"),
    diff_actr1_actr2_copy.TotalNumMentions.alias("total_num_mentions"),
    diff_actr1_actr2_copy.TotalNumSources.alias("total_num_sources"),
    diff_actr1_actr2_copy.TotalNumArticles.alias("total_num_articles"),
    diff_actr1_actr2_copy.TotalAvgTone.alias("total_avg_tone")
    )
join_df = diff_actr1_actr2_copy.join(diff_actr1_actr2 ,(diff_actr1_actr2_copy.CountryCodeA2 ==
    (diff_actr1_actr2_copy.CountryCodeA2 == diff_actr1_actr2.CountryCode))
#Dropping rows with null values
join_df = join_df.na.drop("all",subset=["Actor1Geo_CountryCode","Actor2Geo_CountryCode"])
#To get strings showing relation btwn two countries
concat_asc_codes_udf = udf(concat_asc_codes, StringType())
join_df = join_df.withColumn('Country_Codes_String', concat_asc_codes_udf(join_df.CountryCodeA1, join_df.CountryCodeA2))

join_df = join_df.select(
    join_df.Actor1Geo_CountryCode,
    join_df.Actor1Geo_Country_Name,
    join_df.Actor2Geo_CountryCode,
    join_df.Actor2Geo_Country_Name,
    join_df.EventBaseCode,

```

```

join_df.Event_Base_Code,
join_df.no_of_events,
join_df.total_number_of_events,
join_df.Country_Codes_String,
sum(join_df[c1] for c1 in ["total_number_of_events",
                           "no_of_events"]).alias("sum_events"),
join_df.total_goldsteinscale,
join_df.total_num_mentions,
join_df.total_num_sources,
join_df.total_num_articles,
join_df.total_avg_tone)

#Dropping duplicates based on relation btwn two countries string
join_df = join_df.dropDuplicates(['Country_Codes_String']).sort(join_df.sum_events)

#Mapping event codes with event names
join_df = join_df.join(map_names_event, (join_df.EventBaseCode == map_names_event.EventBaseCode),
                        join_df.Country_Codes_String == map_names_event.Country_Codes_String,
                        join_df.Event_Base_Code == map_names_event.Event_Base_Code,
                        map_names_event.Type_of_Event == join_df.Type1,
                        join_df.Actor1Geo_Country_Name == map_names_event.Actor1Geo_Country_Name,
                        join_df.Actor2Geo_Country_Name == map_names_event.Actor2Geo_Country_Name,
                        join_df.no_of_events == map_names_event.no_of_events,
                        join_df.total_number_of_events == map_names_event.total_number_of_events,
                        join_df.sum_events == map_names_event.sum_events,
                        join_df.total_goldsteinscale == map_names_event.total_goldsteinscale,
                        join_df.total_num_mentions == map_names_event.total_num_mentions,
                        join_df.total_num_sources == map_names_event.total_num_sources,
                        join_df.total_num_articles == map_names_event.total_num_articles,
                        join_df.total_avg_tone == map_names_event.total_avg_tone)

join_df = join_df.join(map_names_event, (join_df.Event_Base_Code == map_names_event.EventBaseCode),
                        join_df.Country_Codes_String == map_names_event.Country_Codes_String,
                        join_df.Actor1Geo_Country_Name == map_names_event.Actor1Geo_Country_Name,
                        join_df.Actor2Geo_Country_Name == map_names_event.Actor2Geo_Country_Name,
                        join_df.EventBaseCode == map_names_event.EventBaseCode,
                        join_df.Event_Base_Code == map_names_event.Event_Base_Code,
                        join_df.Type1 == map_names_event.Type1,
                        map_names_event.Type_of_Event == join_df.Type1,
                        join_df.no_of_events == map_names_event.no_of_events,
                        join_df.total_number_of_events == map_names_event.total_number_of_events,
                        join_df.sum_events == map_names_event.sum_events,
                        join_df.total_goldsteinscale == map_names_event.total_goldsteinscale,
                        join_df.total_num_mentions == map_names_event.total_num_mentions,
                        join_df.total_num_sources == map_names_event.total_num_sources,
                        join_df.total_num_articles == map_names_event.total_num_articles,
                        join_df.total_avg_tone == map_names_event.total_avg_tone)

```

```

    )
    join_df = join_df.sort(join_df.sum_events.desc())
    return join_df

In [44]: def get_state_name(city_state_country,category,country_name):
    tokens = city_state_country.split(',')
    if country_name == 'US':
        if category == 2:
            return tokens[0]
        elif category == 3:
            return tokens[1]
    else :
        if (category == 4) | (category == 5):
            return tokens[1]

def do_state_analysis(df,country_name):
    df_events_with_state = df.select('*').where(df.ActionGeo_CountryCode == country_name)
    df_events_with_state = df_events_with_state.na.drop("all",subset=["ActionGeo_FullName"])
    get_state_name_udf = udf(get_state_name, StringType())
    df_events_with_state = df_events_with_state.withColumn('Action_State', get_state_name_udf(df_events_with_state.ActionGeo_FullName, df_events_with_state.Category))

    df_state_goldstein_events = df_events_with_state.groupBy(df_events_with_state.Action_State).sum().na.drop("all",subset=["Total"])
    df_state_goldstein_events = df_state_goldstein_events.na.drop("all",subset=["Total"])
    df_state_goldstein_events = df_state_goldstein_events.sort(df_state_goldstein_events.Total.desc())
    return df_state_goldstein_events

```

## 2 2015

### 3 AID Analysis

```

In [20]: df_2015 = spark.read.format('CSV').option('sep', '\t').schema(schema).load('inputs/gdpr/2015-aid.csv')

In [107]: df_2015_aid_results = do_aid_analysis(df_2015,'07',df_aids)

In [108]: df_2015_aid_results.limit(100).write.csv('spark-outputs/2015-AID-ANALYSIS',header='true')

```

### 4 DEPORTATION Analysis

```

In [122]: df_2015_deport_results = do_aid_analysis(df_2015,'17',df_deport)

In [123]: df_2015_deport_results = df_2015_deport_results.where(df_2015_deport_results.EventBaseType == 'DEPORTATION')

In [125]: df_2015_deport_results.limit(100).write.csv('spark-outputs/2015-DEPORT-ANALYSIS',header='true')

```

## 5 Top Stories of United States

```
In [136]: top_events_2015_us = df_2015.where((df_2015.Actor1Geo_CountryCode == 'US') |
                                             (df_2015.Actor2Geo_CountryCode == 'US')).sort(df_2015.NumM
top_events_2015_us.select('*').limit(50).write.csv('spark-outputs/2015-TOP-STORIES-US')
```

## 6 Mass Destruction vs Stability of Countries

```
In [182]: mass_destrn_events = df_2015.where(df_2015.EventRootCode == '20').groupBy(df_2015.Ac
fun.sum(df_2015.GoldsteinScale).alias("Total_GoldsteinScale"),
fun.sum(df_2015.AvgTone).alias("Total_AvgTone"),
fun.sum(df_2015.NumMentions).alias("Total_NumMentions"))
mass_destrn_events = mass_destrn_events.sort(mass_destrn_events.Total_GoldsteinScale)
```

```
In [183]: mass_destrn_events = mass_destrn_events.na.drop("all",subset=["ActionGeo_CountryCode"])
```

```
In [184]: mass_destrn_events = mass_destrn_events.join(df_country_codes,mass_destrn_events.Act
```

```
In [186]: mass_destrn_events.limit(50).write.csv('spark-outputs/2015-MASS-DESTRUCTION-VS-STABI
```

## 7 Pair of Countries involved most in Mass Destruction

```
In [244]: df_2015_massdestrn_results = do_aid_analysis(df_2015,'20',df_massdestrn)
```

```
In [245]: df_2015_massdestrn_results = df_2015_massdestrn_results.sort(df_2015_massdestrn_resu
```

```
In [246]: df_2015_massdestrn_results = df_2015_massdestrn_results.na.drop("all",subset=["total
```

```
In [247]: df_2015_massdestrn_results.limit(100).write.csv('spark-outputs/2015-MASS-DESTRN-ANAL
```

## 8 Which State is effecting stability of US more?

```
In [25]: df_2015_state_stability_country_results = do_state_analysis(df_2015,'US')
```

```
In [26]: df_2015_state_stability_country_results.limit(100).write.csv('spark-outputs/2015-STAT
```

## 9 Top Stories of India

```
In [31]: top_events_2015_in = df_2015.where((df_2015.ActionGeo_CountryCode == 'IN')).sort(df_2
top_events_2015_in.select('*').limit(50).write.csv('spark-outputs/2015-TOP-STORIES-IN
```

## 10 Which State is effecting stability of India more?

```
In [45]: df_2015_state_stability_country_results = do_state_analysis(df_2015,'IN')
```

```
In [47]: df_2015_state_stability_country_results.limit(100).write.csv('spark-outputs/2015-STAT
```

## 11 2016

## 12 AID Analysis

```
In [15]: df_2016 = spark.read.format('CSV').option('sep', '\t').schema(schema).load('inputs/gd')
```

```
In [132]: df_2016_aid_results = do_aid_analysis(df_2016, '07', df_aids)
```

```
In [134]: df_2016_aid_results.limit(100).write.csv('spark-outputs/2016-AID-ANALYSIS', header='t')
```

## 13 DEPORTATION Analysis

```
In [126]: df_2016_deport_results = do_aid_analysis(df_2016, '17', df_deport)
```

```
In [127]: df_2016_deport_results = df_2016_deport_results.where(df_2016_deport_results.EventBas
```

```
In [128]: df_2016_deport_results.limit(100).write.csv('spark-outputs/2016-DEPORT-ANALYSIS', head
```

## 14 Top Stories of United States

```
In [137]: top_events_2016_us = df_2016.where((df_2016.Actor1Geo_CountryCode == 'US') |  
                                             (df_2016.Actor2Geo_CountryCode == 'US')).sort(df_2016.NumM  
top_events_2016_us.select('*').limit(50).write.csv('spark-outputs/2016-TOP-STORIES-US')
```

## 15 Mass Destruction vs Stability of Countries

```
In [187]: mass_destrn_events_2016 = df_2016.where(df_2016.EventRootCode == '20').groupBy(df_20  
                                             fun.sum(df_2016.GoldsteinScale).alias("Total_GoldsteinScale"),  
                                             fun.sum(df_2016.AvgTone).alias("Total_AvgTone"),  
                                             fun.sum(df_2016.NumMentions).alias("Total_NumMentions"))  
mass_destrn_events_2016 = mass_destrn_events_2016.sort(mass_destrn_events_2016.Total
```

```
In [188]: mass_destrn_events_2016 = mass_destrn_events_2016.na.drop("all", subset=["ActionGeo_Co
```

```
In [189]: mass_destrn_events_2016 = mass_destrn_events_2016.join(df_country_codes, mass_destrn_
```

```
In [190]: mass_destrn_events_2016.limit(50).write.csv('spark-outputs/2016-MASS-DESTRUCTION-VS-
```

## 16 Pair of Countries involved most in Mass Destruction

```
In [249]: df_2016_massdestrn_results = do_aid_analysis(df_2016, '20', df_massdestrn)
```

```
In [250]: df_2016_massdestrn_results = df_2016_massdestrn_results.sort(df_2016_massdestrn_resu
```

```
In [251]: df_2016_massdestrn_results = df_2016_massdestrn_results.na.drop("all", subset=["total
```

```
In [252]: df_2016_massdestrn_results.limit(100).write.csv('spark-outputs/2016-MASS-DESTRN-ANAL
```

**17 Which state is effecting stability of US more?**

```
In [27]: df_2016_state_stability_country_results = do_state_analysis(df_2016, 'US')
```

```
In [28]: df_2016_state_stability_country_results.limit(100).write.csv('spark-outputs/2016-STATI
```

## 18 Top Stories of India

```
In [48]: top_events_2016_in = df_2016.where((df_2016.ActionGeo_CountryCode == 'IN')).sort(df_2016.EventCount, ascending=False)
top_events_2016_in.select('*').limit(50).write.csv('spark-outputs/2016-TOP-STORIES-IN')
```

**19 Which State is effecting stability of India more?**

```
In [49]: df_2016_state_stability_country_results_in = do_state_analysis(df_2016, 'IN')
```

```
In [50]: df_2016_state_stability_country_results_in.limit(100).write.csv('spark-outputs/2016-S'
```

## 20 Scrapbook

```
In [179]: df_country_codes.take(7)
```

```
Out[179]: [Row(Country_Name='AF', Nato_Country_Code='Afghanistan'),
Row(Country_Name='AX', Nato_Country_Code='Akrotiri'),
Row(Country_Name='AL', Nato_Country_Code='Albania'),
Row(Country_Name='AG', Nato_Country_Code='Algeria'),
Row(Country_Name='AQ', Nato_Country_Code='American Samoa'),
Row(Country_Name='AN', Nato_Country_Code='Andorra'),
Row(Country_Name='AO', Nato_Country_Code='Angola')]
```

```
In [55]: df.take(1)
```

```
Out[55]: [Row(GLOBALEVENTID=597122373, SQLDATE=20151110, MonthYear=201511, Year=2015, FractionM
```

```
In [56]: df.count()
```

Out[56]: 73270827

```
In [58]: df 2015.take(1)
```

```
Out[58]: [Row(GLOBALEVENTID=478037761, SQLDATE=20051025, MonthYear=200510, Year=2005, FractionM
```

```
In [59]: df_2015.count()
```

Out [59]: 66370819

```
In [64]: df_2015.columns
```



```
Out[64]: ['GLOBALEVENTID',
          'SQLDATE',
          'MonthYear',
          'Year',
          'FractionDate',
          'Actor1Code',
          'Actor1Name',
          'Actor1CountryCode',
          'Actor1KnownGroupCode',
          'Actor1EthnicCode',
          'Actor1Religion1Code',
          'Actor1Religion2Code',
          'Actor1Type1Code',
          'Actor1Type2Code',
          'Actor1Type3Code',
          'Actor2Code',
          'Actor2Name',
          'Actor2CountryCode',
          'Actor2KnownGroupCode',
          'Actor2EthnicCode',
          'Actor2Religion1Code',
          'Actor2Religion2Code',
          'Actor2Type1Code',
          'Actor2Type2Code',
          'Actor2Type3Code',
          'IsRootEvent',
          'EventCode',
          'EventBaseCode',
          'EventRootCode',
          'QuadClass',
          'GoldsteinScale',
          'NumMentions',
          'NumSources',
          'NumArticles',
          'AvgTone',
          'Actor1Geo_Type',
          'Actor1Geo_FullName',
          'Actor1Geo_CountryCode',
          'Actor1Geo_ADM1Code',
          'Actor1Geo_Lat',
          'Actor1Geo_Long',
          'Actor1Geo_FeatureID',
          'Actor2Geo_Type',
          'Actor2Geo_FullName',
          'Actor2Geo_CountryCode',
          'Actor2Geo_ADM1Code',
          'Actor2Geo_Lat',
          'Actor2Geo_Long',
```

```

'Actor2Geo_FeatureID',
'ActionGeo_Type',
'ActionGeo_FullName',
'ActionGeo_CountryCode',
'ActionGeo_ADM1Code',
'ActionGeo_Lat',
'ActionGeo_Long',
'ActionGeo_FeatureID',
'DATEADDED',
'SOURCEURL']

```

## 21 Mass Destruction - Israel Analysis

```

In [165]: mass_destrn_events = df_2015.where((df_2015.EventRootCode == '20') & (df_2015.Action
mass_destrn_events = mass_destrn_events.sort(mass_destrn_events.GoldsteinScale.asc())
mass_destrn_events.take(5)

```

```

Out[165]: [Row(GLOBALEVENTID=482014062, SQLDATE=20151105, MonthYear=201511, Year=2015, Fraction
Row(GLOBALEVENTID=482036371, SQLDATE=20151105, MonthYear=201511, Year=2015, Fraction
Row(GLOBALEVENTID=482036710, SQLDATE=20151105, MonthYear=201511, Year=2015, Fraction
Row(GLOBALEVENTID=482036720, SQLDATE=20151105, MonthYear=201511, Year=2015, Fraction
Row(GLOBALEVENTID=482059722, SQLDATE=20151105, MonthYear=201511, Year=2015, Fraction

```

```

In [68]: top_events_2015_india = df_2015.where((df_2015.Actor1CountryCode == 'IND') |
(df_2015.Actor2CountryCode == 'IND')).sort(df_2015.NumMention
top_events_2015_india.select(top_events_2015_india.Actor1Name,
top_events_2015_india.Actor2Name,
top_events_2015_india.NumMentions,
top_events_2015_india.NumSources,
top_events_2015_india.AvgTone,
top_events_2015_india.SOURCEURL).show(n=5)

```

```

+-----+-----+-----+-----+-----+-----+
|Actor1Name|Actor2Name|NumMentions|NumSources|    AvgTone|          SOURCEURL|
+-----+-----+-----+-----+-----+-----+
|      null|    KERALA|      2730|      248|   -4.55655|http://www.njhera...|
|  KOLKATA|  NEPALESE|      2473|      220|   -3.4093997|http://www.washin...|
| NEW DELHI| NATIONALS|      2316|      499|   -3.7399256|http://www.nwitim...|
|    KERALA|    KIRKUK|      2316|        2|   -4.648082|http://www.arabhe...|
|    POLICE|    DELHI|      2265|      201|  -11.135494|http://news.yahoo...|
+-----+-----+-----+-----+-----+-----+

```

only showing top 5 rows

```

In [69]: top_events_2016_india = df.where((df.Actor1CountryCode == 'IND') |
(df.Actor2CountryCode == 'IND')).select(*df.columns).sort(df.NumMention
top_events_2016_india.select(top_events_2016_india.Actor1Name,

```

```
top_events_2016_india.Actor2Name,
top_events_2016_india.NumMentions,
top_events_2016_india.NumSources,
top_events_2016_india.AvgTone,
top_events_2016_india.SOURCEURL).show(n=5)
```

```
+-----+-----+-----+-----+-----+-----+
| Actor1Name | Actor2Name | NumMentions | NumSources | AvgTone | SOURCEURL |
+-----+-----+-----+-----+-----+-----+
| MANMOHAN SINGH | MINIST FOR HOME A... | 4710 | 3 | -2.630281 | http://aninews.in |
| SRINAGAR | COMMANDANT | 4652 | 1 | -1.9250137 | http://aninews.in |
| MINIST FOR HOME A... | MANMOHAN SINGH | 4148 | 3 | -2.654616 | http://aninews.in |
| NEW DELHI | HOME MINIST | 3070 | 5 | -2.1281862 | http://aninews.in |
| NEW DELHI | CONGRESS | 2896 | 1 | -1.9349867 | http://aninews.in |
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
In [75]: top_events_2015_india.select(top_events_2015_india.Actor1Name,
top_events_2015_india.Actor2Name,
top_events_2015_india.NumMentions,
top_events_2015_india.NumSources,
top_events_2015_india.AvgTone,
top_events_2015_india.SOURCEURL).limit(5).write.csv('top_
```

```
In [76]: top_events_2016_india.select(top_events_2016_india.Actor1Name,
top_events_2016_india.Actor2Name,
top_events_2016_india.NumMentions,
top_events_2016_india.NumSources,
top_events_2016_india.AvgTone,
top_events_2016_india.SOURCEURL).limit(5).write.csv('top_
```

```
In [78]: top_events_2015_india.sample(False,0.1).select('*').limit(100).write.csv('top_events_'
```

```
In [83]: df_2015.sample(False,0.05).select('*').where(df_2015.EventRootCode == '07').limit(100)
```

```
In [103]: import pyspark.sql.functions as fun
df_2015.sample(False,0.05).select('*').where(df_2015.EventRootCode == '07').groupBy(
df_2015.Actor2Geo_CountryCode == df_country_codes.Nato_Count
```

```
In [116]: df_2015.select('*').where(df_2015.EventCode == '174').take(5)
```

```
Out[116]: [Row(GLOBALEVENTID=478039312, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478041233, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478041234, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478041235, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478041236, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
```

```

In [104]: df_2015.select('*').where(df_2015.EventRootCode == '07').groupBy(df_2015.Actor2Geo_C
df_2015.Actor2Geo_CountryCode == df_country_codes.Nato_Count

In [105]: df_2015.select('*').where(df_2015.EventRootCode == '07').groupBy(df_2015.Actor1Geo_C
df_2015.Actor1Geo_CountryCode == df_country_codes.Nato_Count

In [152]: mass_destrn_events = df_2015.where(df_2015.EventRootCode == '20')
mass_destrn_events.select('*').take(5)

Out[152]: [Row(GLOBALEVENTID=478060576, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478060581, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478070429, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478082947, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction
Row(GLOBALEVENTID=478101647, SQLDATE=20151023, MonthYear=201510, Year=2015, Fraction

```