## Importing the necessary libraries

In [1]:

```
import pandas as pd
import numpy as np
import os
os.chdir('C:\\Users\\Swapnil bandekar\\Downloads\\Swapnil\\Data Analytics\\My Work\\Python\
```

## Reading the file

In [10]:

```
terror = pd.read_csv('terror.csv', encoding='latin')
terror.head()
```

```
C:\Users\Swapnil bandekar\Downloads\Swapnil\Data Analytics\anaconda3\lib\sit
e-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (4,6
1,62,66,116,117,123) have mixed types.Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[10]:

| | eventid | iyear | imonth | iday | approxdate | extended | resolution | country | country_txt | r |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.970000e+11 | 1970 | 0 | 0 | NaN | 0 | NaN | 58 | Dominican Republic | |
| **1** | 1.970000e+11 | 1970 | 0 | 0 | NaN | 0 | NaN | 130 | Mexico | |
| **2** | 1.970010e+11 | 1970 | 1 | 0 | NaN | 0 | NaN | 160 | Philippines | |
| **3** | 1.970010e+11 | 1970 | 1 | 0 | NaN | 0 | NaN | 78 | Greece | |
| **4** | 1.970010e+11 | 1970 | 1 | 0 | NaN | 0 | NaN | 101 | Japan | |

5 rows × 137 columns

In [11]:

```
terror.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156772 entries, 0 to 156771
Columns: 137 entries, eventid to related
dtypes: float64(56), int64(23), object(58)
memory usage: 163.9+ MB
```

In [12]:

```
terror.dtypes
```

Out[12]:

```
eventid       float64
iyear           int64
imonth          int64
iday            int64
approxdate     object
               ...
INT_LOG         int64
INT_IDEO        int64
INT_MISC        int64
INT_ANY         int64
related        object
Length: 137, dtype: object
```

In [14]:

```
terror.describe(include='all')
```

Out[14]:

| | eventid | iyear | imonth | iday | approxdate | extended |
|---|---|---|---|---|---|---|
| **count** | 1.567720e+05 | 156772.000000 | 156772.000000 | 156772.000000 | 4756 | 156772.000000 |
| **unique** | NaN | NaN | NaN | NaN | 1426 | NaN |
| **top** | NaN | NaN | NaN | NaN | July 1-14, 2014 | NaN |
| **freq** | NaN | NaN | NaN | NaN | 27 | NaN |
| **mean** | 2.000539e+11 | 2000.474083 | 6.484666 | 15.455215 | NaN | 0.041347 |
| **std** | 1.298281e+09 | 12.982397 | 3.392225 | 8.815533 | NaN | 0.199091 |
| **min** | 1.970000e+11 | 1970.000000 | 0.000000 | 0.000000 | NaN | 0.000000 |
| **25%** | 1.989080e+11 | 1989.000000 | 4.000000 | 8.000000 | NaN | 0.000000 |
| **50%** | 2.005070e+11 | 2005.000000 | 6.000000 | 15.000000 | NaN | 0.000000 |
| **75%** | 2.013060e+11 | 2013.000000 | 9.000000 | 23.000000 | NaN | 0.000000 |
| **max** | 2.015120e+11 | 2015.000000 | 12.000000 | 31.000000 | NaN | 1.000000 |

11 rows × 137 columns

In [17]:

```
print(terror.columns.tolist())
```

```
['eventid', 'iyear', 'imonth', 'iday', 'approxdate', 'extended', 'resolutio
n', 'country', 'country_txt', 'region', 'region_txt', 'provstate', 'city',
'latitude', 'longitude', 'specificity', 'vicinity', 'location', 'summary',
'crit1', 'crit2', 'crit3', 'doubtterr', 'alternative', 'alternative_txt', 'm
ultiple', 'success', 'suicide', 'attacktype1', 'attacktype1_txt', 'attacktyp
e2', 'attacktype2_txt', 'attacktype3', 'attacktype3_txt', 'targtype1', 'targ
type1_txt', 'targsubtype1', 'targsubtype1_txt', 'corp1', 'target1', 'natlty
1', 'natlty1_txt', 'targtype2', 'targtype2_txt', 'targsubtype2', 'targsubtyp
e2_txt', 'corp2', 'target2', 'natlty2', 'natlty2_txt', 'targtype3', 'targtyp
e3_txt', 'targsubtype3', 'targsubtype3_txt', 'corp3', 'target3', 'natlty3',
'natlty3_txt', 'gname', 'gsubname', 'gname2', 'gsubname2', 'gname3', 'ingrou
p', 'ingroup2', 'ingroup3', 'gsubname3', 'motive', 'guncertain1', 'guncertai
n2', 'guncertain3', 'nperps', 'nperpcap', 'claimed', 'claimmode', 'claimmode
_txt', 'claim2', 'claimmode2', 'claimmode2_txt', 'claim3', 'claimmode3', 'cl
aimmode3_txt', 'compclaim', 'weaptype1', 'weaptype1_txt', 'weapsubtype1', 'w
eapsubtype1_txt', 'weaptype2', 'weaptype2_txt', 'weapsubtype2', 'weapsubtype
2_txt', 'weaptype3', 'weaptype3_txt', 'weapsubtype3', 'weapsubtype3_txt', 'w
eaptype4', 'weaptype4_txt', 'weapsubtype4', 'weapsubtype4_txt', 'weapdetai
l', 'nkill', 'nkillus', 'nkillter', 'nwound', 'nwoundus', 'nwoundte', 'prope
rty', 'propextent', 'propextent_txt', 'propvalue', 'propcomment', 'ishostki
d', 'nhostkid', 'nhostkidus', 'nhours', 'ndays', 'divert', 'kidhijcountry',
'ransom', 'ransomamt', 'ransomamtus', 'ransompaid', 'ransompaidus', 'ransomn
ote', 'hostkidoutcome', 'hostkidoutcome_txt', 'nreleased', 'addnotes', 'scit
e1', 'scite2', 'scite3', 'dbsource', 'INT_LOG', 'INT_IDEO', 'INT_MISC', 'INT
_ANY', 'related']
```

## Q.1) How many attacks happened in India?

Hint: Use the country_txt column.

In [21]:

```
# Solution 1

terror.query(" country_txt == 'India'").shape[0]
```

Out[21]:

```
9940
```

In [23]:

```
# Solution 2

terror.query(" country_txt == 'India'")['country_txt'].value_counts()
```

Out[23]:

```
India    9940
Name: country_txt, dtype: int64
```

In [ ]:

```
# Solution 3

data[data['country_txt']=="India"].shape[0]
```

## Q.2) How many attacks happened in India and upto 3 people were killed?

Hint: Use the country_txt and nkill column.

In [27]:

```
# Solution 1

terror.query(" country_txt=='India' & nkill < 4").shape[0]
```

Out[27]:

8362

In [ ]:

```
# Solution 2

data[(data['country_txt']=='India')&(data['nkill']<=3)].shape[0]
```

## Q.3) Extract the city and summary for attacks above

Hint: Use country_txt, nkill and summary columns

In [31]:

```
terror.query(" country_txt=='India' & nkill < 4")[['city','summary']]
```

Out[31]:

| | city | summary |
|---|---|---|
| **1185** | New Delhi | NaN |
| **3780** | New Delhi | NaN |
| **5251** | Bombay | NaN |
| **7266** | Imphal | NaN |
| **8609** | Unknown | NaN |
| **...** | ... | ... |
| **156695** | Terem | 12/29/2015: Assailants abducted three students... |
| **156698** | Peravurani | 12/29/2015: An explosive device was discovered... |
| **156713** | Tulsibari | 12/29/2015: Assailants abducted Hazrat Ali in ... |
| **156725** | Zhutovi | 12/30/2015: Assailants attempted to extort mon... |
| **156756** | Srinagar | 12/31/2015: Assailants threw a grenade at an I... |

8362 rows × 2 columns

## Q.4) In a single terror incident in India, find out top 5 cities by number killed

Hint: Use each row as as a unique terror incident. Use country_txt, nkill and city columns

In [196]:

```
terror.query("country_txt=='India'").sort_values('nkill',ascending=False)[['city','nkill','
```

Out[196]:

| | city | nkill | iyear |
|---|---|---|---|
| **81000** | Mumbai | 188.0 | 2006 |
| **96598** | Jhargam | 115.0 | 2010 |
| **54339** | Bombay | 115.0 | 1992 |
| **95860** | Dantewada district | 82.0 | 2010 |
| **56837** | Banabari | 70.0 | 1994 |

## Q.5) In a single terror incident in India, find out top 5 cities by number killed and wounded

Hint: Use each row as as a unique terror incident. Use country_txt, nkill,nwound and city columns

In [207]:

```
terror.query("country_txt=='India'").sort_values('nkill',ascending=False)[['city','nkill','
```

Out[207]:

| | city | nkill | nwound | iyear |
|---|---|---|---|---|
| **81000** | Mumbai | 188.0 | 817.0 | 2006 |
| **96598** | Jhargam | 115.0 | 140.0 | 2010 |
| **54339** | Bombay | 115.0 | 0.0 | 1992 |
| **95860** | Dantewada district | 82.0 | 0.0 | 2010 |
| **56837** | Banabari | 70.0 | 100.0 | 1994 |

In [208]:

```
terror.query("country_txt=='India'").sort_values('nwound',ascending=False)[['city','nkill',
```

Out[208]:

| | city | nkill | nwound | iyear |
|---|---|---|---|---|
| **81000** | Mumbai | 188.0 | 817.0 | 2006 |
| **89128** | Dispur | 37.0 | 235.0 | 2008 |
| **30378** | Chennai | 32.0 | 200.0 | 1987 |
| **79045** | New Delhi | 55.0 | 155.0 | 2005 |
| **75769** | Mumbai | 52.0 | 150.0 | 2003 |

## Q.6) How many attacks were successful that were suicide attacks?

Hint: Use suicide and success columns

In [40]:

```
print(terror['suicide'].dtypes)
print(terror['suicide'].unique())
print(terror['success'].dtypes)
print(terror['success'].unique())
```

```
int64
[0 1]
int64
[1 0]
```

In [41]:

```
terror.query(" suicide==1 & success==1 ").shape[0]
```

Out[41]:

```
4260
```

## Q.7) Label all the incidents where the number killed was more than 5 as severe.

Hint: Use Aggregations and manipulations using apply and map

#map: map a function to each element of a series object

#Suppose we want to label all the incidents where the number killed was more than 5 as severe. This would involve applying a function on each element of the series, map helps in doing that.

#You can use lambda functions as well (if else follows a special form when used with lambdas ) Use the nkill column.

In [45]:

```
# Solution 1

def sev(x):
    y = 'severe' if x > 5 else 'Not severe'
    return y
```

In [46]:

```
terror['severity'] = terror['nkill'].apply(sev)
```

In [61]:

```
# Solution 2

terror['severity_1'] = pd.Series(map(lambda x : 'severe' if x>5 else 'Not severe',terror['n
```

In [235]:

```
def get_label(x):
    if x>5:
        return 'Severe'
    else:
        return 'Not Severe'
data['nkill'].map(get_label)
```

Out[235]:

```
0         Not Severe
1         Not Severe
2         Not Severe
3         Not Severe
4         Not Severe
             ...
156767    Not Severe
156768    Not Severe
156769    Not Severe
156770    Not Severe
156771    Not Severe
Name: nkill, Length: 156772, dtype: object
```

## Q.8) write a function to label an incident that was both successful and suicidal

Hint: We can use apply to use a function column wise where you can use columns success and suicide in the data.

In [105]:

```
# Solution 1

def ss(cols):
    x = cols[0]
    y = cols[1]
    z = 'Suicide Successful' if x==1 & y==1 else 'Other'
    return z
```

In [110]:

```
terror['sui_success'] = terror[['suicide','success']].apply(ss,axis=1)
```

In [111]:

```
terror['sui_success'].unique()
```

Out[111]:

```
array(['Other', 'Suicide Successful'], dtype=object)
```

In [119]:

```
terror[terror['sui_success']== 'Suicide Successful'].shape[0]
```

Out[119]:

```
4260
```

In [122]:

```
# Solution 2

terror['sui_success_1']=pd.Series(map(lambda x,y : 'Suicide Successful' if x==1 & y==1 else
```

In [ ]:

```
# Alternate Solutions

# 3

def get_label(row):
    if row['success']==1 and row['suicide']==1:
        return 1
    else:
        return 0
terror.apply(get_label,axis=1)
terror.apply(get_label,axis=1).unique()

# 4

terror['b']=terror.apply(lambda row: 1 if row['success']==1 and row['suicide']==1 else 0,ax
```

## Q.9) Create a new category representing if the incident occured in Afghanistan,Pakistan or India as one level of the category and all the other countries as another level. Label all other countries as ROW and new column which contains the new category as 'Local' in the data.

Hint: Create a new category representing if the incident occured in Afghanistan,Pakistan or India as one level of the category and all the other countries as another level. Use the columns – country_txt and create a new column called "Local" in the dataset.

In [138]:

```
# Solution 1

def cat(x):
    y = 'Local' if x=='Afghanistan' or x=='India' or x=='Pakistan' else 'ROW'
    return y
```

In [139]:

```
terror['New_Category'] = terror['country_txt'].apply(cat)
```

In [141]:

```
terror['New_Category'].unique()
```

Out[141]:

```
array(['ROW', 'Local'], dtype=object)
```

In [158]:

```
terror.query(" New_Category =='Local'")['country_txt'].unique()
```

Out[158]:

```
array(['Pakistan', 'India', 'Afghanistan'], dtype=object)
```

In [160]:

```
# Solution 2

terror['New_Category_1'] = pd.Series(map(lambda x : 'Local' if x=='Afghanistan' or x=='Indi
```

In [161]:

```
terror.query(" New_Category_1 =='Local'")['country_txt'].unique()
```

Out[161]:

```
array(['Pakistan', 'India', 'Afghanistan'], dtype=object)
```

In [ ]:

```
# Alternate Solutions

# 3

def get_label(row):
    if row['country_txt']=='India' or row['country_txt']=='Afghanistan' or row['country_txt
        return 'Af-Pak-India'
    else:
        return 'ROW'
terror.apply(get_label,axis=1)

terror['Local']=terror.apply(get_label,axis=1)


# 4

terror.apply(lambda row: 'Af-Pak-India' if row['country_txt']=='India' or row['country_txt'
```

## Q.10) How many incidents happened in Af-Pak-India vs ROW?

Hint: Use the newly created column “Local” and do a value_count or use the size function.

In [168]:

```
# Solution 1

terror['New_Category'].value_counts()
```

Out[168]:

```
ROW      124374
Local     32398
Name: New_Category, dtype: int64
```

In [172]:

```
# Solution 2

terror.groupby('New_Category').size()
```

Out[172]:

```
New_Category
Local     32398
ROW      124374
dtype: int64
```

## Q.11) List the number of suicides attacks and average kills by Af-Pak-India vs ROW. Rename columns in the output as Average_Kills for average kills and Number_Incidents for count of suicide attacks.

Hint: Use the newly created Local column and Suicide column for group by summary and use the nkill column to take average and count.

In [181]:

```
# Solution 1

terror.groupby(['New_Category','suicide']).agg({'nkill':[np.mean,np.size]}).rename(columns=
```

Out[181]:

| | | nkill | |
|---|---|---|---|
| | | Average_kills | Number_Incidents |
| New_Category | suicide | | |
| Local | 0 | 1.816514 | 30936.0 |
| | 1 | 8.127404 | 1462.0 |
| ROW | 0 | 2.167010 | 121065.0 |
| | 1 | 11.511403 | 3309.0 |

In [192]:

```
# Solution 2

terror.pivot_table(index='New_Category',columns='suicide',values='nkill',aggfunc=[np.mean,n
```

Out[192]:

| | Average_kills | | Number_Incidents | |
|---|---|---|---|---|
| suicide | 0 | 1 | 0 | 1 |
| New_Category | | | | |
| Local | 1.816514 | 8.127404 | 30936.0 | 1462.0 |
| ROW | 2.167010 | 11.511403 | 121065.0 | 3309.0 |