Cyber Security Breaches using Temporal Analysis, Network Analysis, Categorical Analysis, Anomaly Detection and Heatmaps

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sys
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
```

Dataset 1

```
df = pd.read_csv("df_1.csv")
```

```
df.head()
```

	Unnamed: 0	Entity	Year	Records	Organization type	Method	Sources
0	0	21st Century Oncology	2016	2200000	healthcare	hacked	[5][6]
1	1	500px	2020	14870304	social networking	hacked	[7]
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security	[8][9]
3	3	Adobe Systems Incorporated	2013	152000000	tech	hacked	[10]
4	4	Adobe Inc.	2019	7500000	tech	poor security	[11][12]

```
df.shape
```

```
(352, 7)
```

```
df.dtypes
```

```
Unnamed: 0            int64
Entity               object
Year                 object
Records              object
Organization type    object
Method               object
Sources              object
dtype: object
```

```
df.drop(['Sources'], axis=1, inplace=True)
```

```
df.isnull().any()
```

```
Unnamed: 0           False
Entity               False
Year                 False
Records               True
Organization type    False
Method                True
dtype: bool
```

```
df.isnull().sum()
```

```
Unnamed: 0           0
Entity               0
Year                 0
Records              2
Organization type    0
Method               1
dtype: int64
```

```
df.columns = ['id', 'Entity', 'Year', 'Records', 'Organization type', 'Method']
```

```
df.head(10)
```

	id	Entity	Year	Records	Organization type	Method
0	0	21st Century Oncology	2016	2200000	healthcare	hacked
1	1	500px	2020	14870304	social networking	hacked
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security
3	3	Adobe Systems Incorporated	2013	152000000	tech	hacked
4	4	Adobe Inc.	2019	7500000	tech	poor security
5	5	Advocate Medical Group	2017	4000000	healthcare	lost / stolen media
6	6	AerServ (subsidiary of InMobi)	2018	75000	advertising	hacked
7	7	Affinity Health Plan, Inc.	2013	344579	healthcare	lost / stolen media
8	8	Airtel	2019	320000000	telecommunications	poor security
9	9	Air Canada	2018	20000	transport	hacked

```
table_year_df = df['Year'].value_counts()
table_year_df
```

```
2011             34
2020             31
2019             30
2015             28
2013             28
2018             26
2014             25
2012             23
2016             22
2010             19
2008             16
2021             13
2009             13
2007             12
2017              9
2006              7
2005              6
2022              5
2004              2
2019-2020         1
2018-2019         1
2014 and 2015     1
Name: Year, dtype: int64
```

```
df['Year'] = df['Year'].astype(str)
df['Year'] = df['Year'].str[:4]
df['Year'] = df['Year'].astype(int)
```

```
df['Year'].head()
```

```
0    2016
1    2020
2    2020
3    2013
4    2019
Name: Year, dtype: int64
```

```
df.dtypes
```

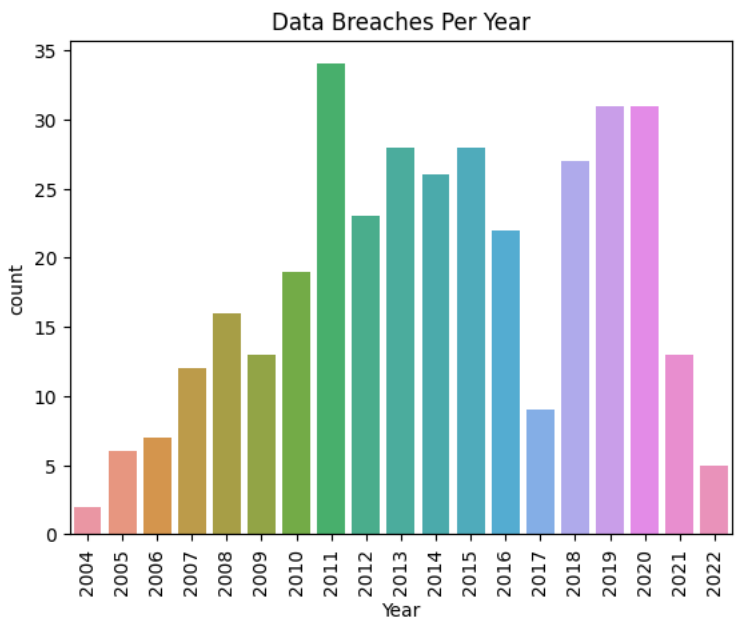
```
id                    int64
Entity               object
Year                  int64
Records              object
Organization type    object
Method               object
dtype: object
```

```
table_year_df = df['Year'].value_counts()
table_year_df
```

```
2011    34
2019    31
2020    31
```

```
2015    28
2013    28
2018    27
2014    26
2012    23
2016    22
2010    19
2008    16
2021    13
2009    13
2007    12
2017     9
2006     7
2005     6
2022     5
2004     2
Name: Year, dtype: int64
```

```
sns.countplot(x='Year', data=df);
plt.title('Data Breaches Per Year')
plt.xticks(rotation=90);
```

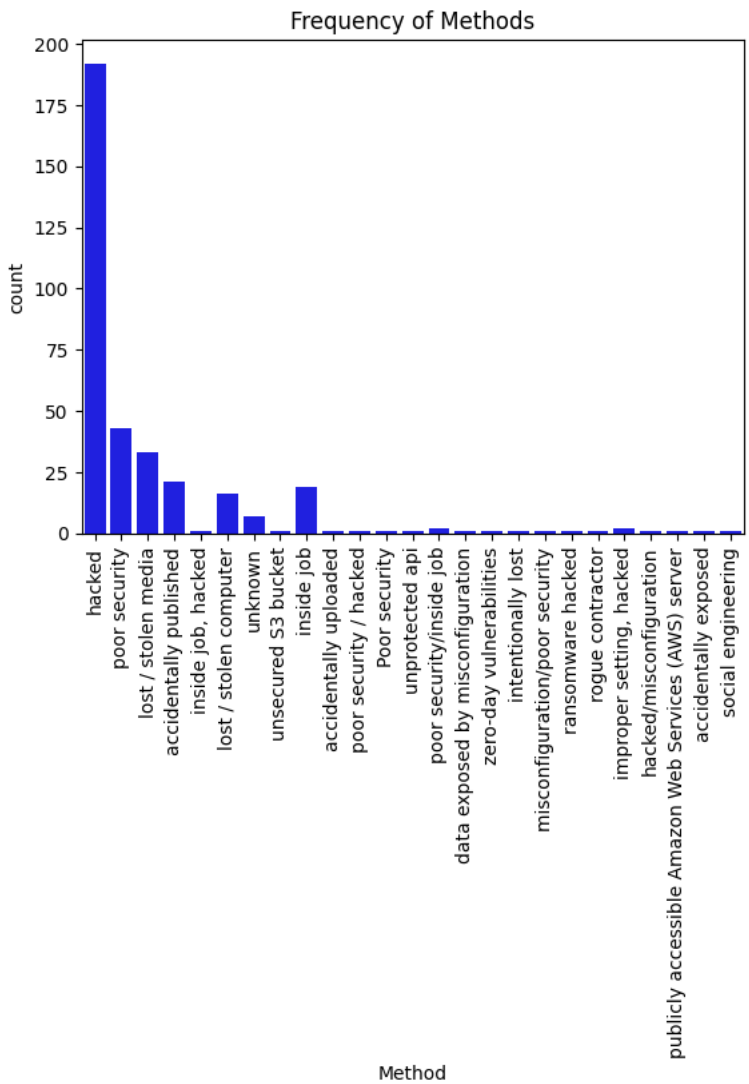



```
sns.countplot(x='Year', data=df, order=table_year_df.index.values);
plt.title('Data Breaches in Order Desc')
plt.xticks(rotation=90);
```

```
table1 = df['Method'].value_counts()
table1
```

```
hacked                                                    192
poor security                                              43
lost / stolen media                                        33
accidentally published                                     21
inside job                                                 19
lost / stolen computer                                     16
unknown                                                     7
improper setting, hacked                                    2
poor security/inside job                                    2
intentionally lost                                          1
accidentally exposed                                        1
publicly accessible Amazon Web Services (AWS) server        1
hacked/misconfiguration                                     1
rogue contractor                                            1
ransomware hacked                                           1
misconfiguration/poor security                              1
unprotected api                                             1
zero-day vulnerabilities                                    1
data exposed by misconfiguration                            1
Poor security                                               1
poor security / hacked                                      1
accidentally uploaded                                       1
unsecured S3 bucket                                         1
inside job, hacked                                          1
social engineering                                          1
Name: Method, dtype: int64
```

```
sns.countplot(x='Method', data=df,color="blue");
plt.title('Frequency of Methods')
plt.xticks(rotation=90);
```

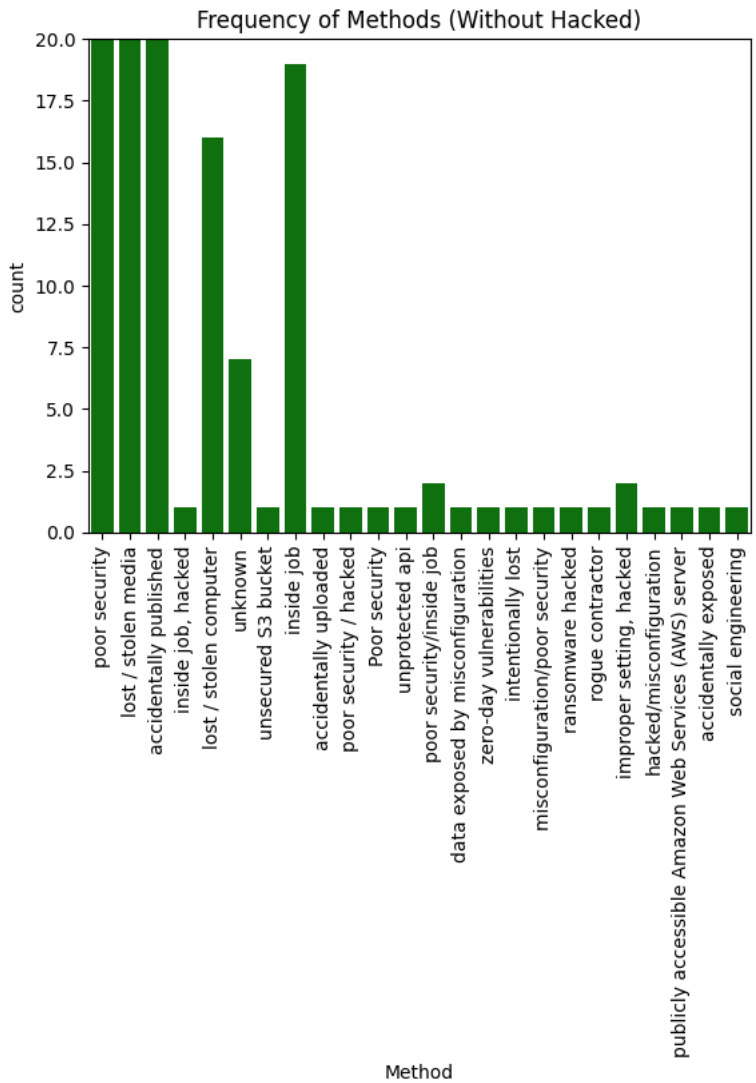


```
df_nothacked = df.loc[df['Method'] != 'hacked']
df_nothacked.head()
```

	id	Entity	Year	Records	Organization type	Method
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security
4	4	Adobe Inc.	2019	7500000	tech	poor security
5	5	Advocate Medical Group	2017	4000000	healthcare	lost / stolen media
7	7	Affinity Health Plan, Inc.	2013	344579	healthcare	lost / stolen media
8	8	Airtel	2019	320000000	telecommunications	poor security

```
sns.countplot(x='Method', data=df_nothacked,color="green");
plt.title('Frequency of Methods (Without Hacked)')
plt.xticks(rotation=90);
plt.ylim([0,20])
```

```
(0.0, 20.0)
```




```
table2 = df['Organization type'].value_counts()
table2.head(23)
```

```
web                     53
healthcare              47
financial               38
government              30
retail                  27
tech                    19
academic                13
telecoms                12
```

```
gaming                    12
social network             8
hotel                      8
transport                  7
military                   7
energy                     4
restaurant                 3
media                      3
mobile carrier             2
social media               2
government, military       2
telecom                    2
tech, retail               2
government, healthcare     2
telecommunications         2
Name: Organization type, dtype: int64
```

```
org_counts = df['Organization type'].value_counts().rename('org_counts')

df_org = df.merge(org_counts.to_frame(),
                  left_on='Organization type',
                  right_index=True)
```

```
org_counts.head()
```

```
web           53
healthcare    47
financial     38
government    30
retail        27
Name: org_counts, dtype: int64
```

```
df_org.head()
```

	id	Entity	Year	Records	Organization type	Method	org_counts
0	0	21st Century Oncology	2016	2200000	healthcare	hacked	47
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security	47
5	5	Advocate Medical Group	2017	4000000	healthcare	lost / stolen media	47
7	7	Affinity Health Plan, Inc.	2013	344579	healthcare	lost / stolen media	47
14	14	Ankle & Foot Center of Tampa Bay, Inc.	2021	156000	healthcare	hacked	47

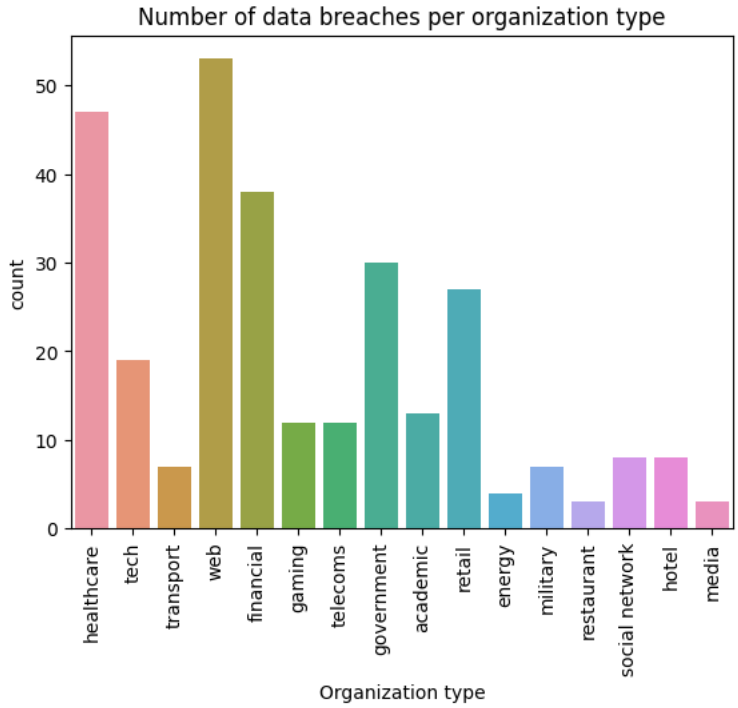
```
df_org_upper = df_org[df_org.org_counts > 2]
```

```
df_org_upper
```

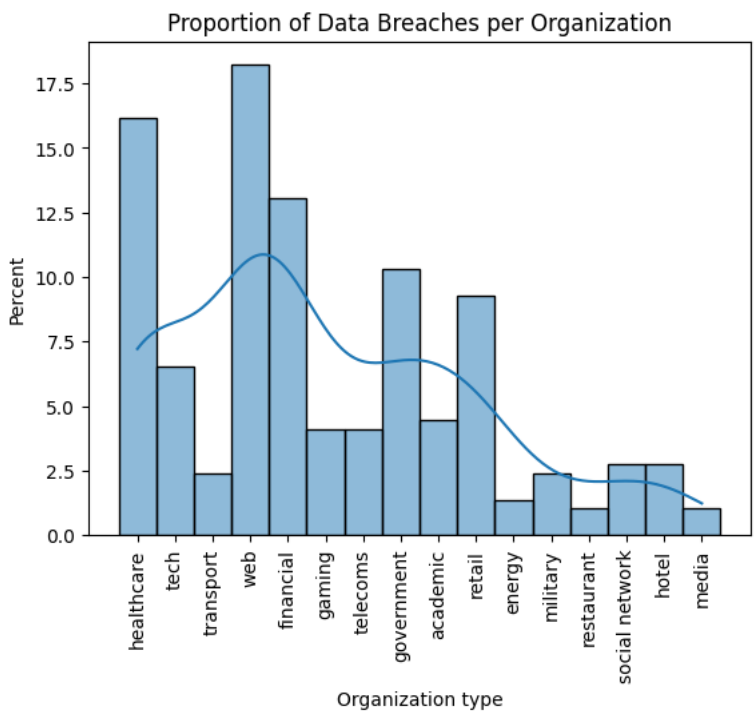
	id	Entity	Year	Records	Organization type	Method	org_counts
0	0	21st Century Oncology	2016	2200000	healthcare	hacked	47
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security	47
5	5	Advocate Medical Group	2017	4000000	healthcare	lost / stolen media	47
7	7	Affinity Health Plan, Inc.	2013	344579	healthcare	lost / stolen media	47
14	14	Ankle & Foot Center of Tampa Bay, Inc.	2021	156000	healthcare	hacked	47
...	...	...	...	...	...	...	...
260	260	Starwoodincluding Westin Hotels & Resorts and ...	2015	54 locations	hotel	hacked	8
286	286	Trump Hotels	2014	8 locations	hotel	hacked	8
211	211	Nippon Television	2016	430000	media	hacked	3
251	251	Sony Pictures	2014	100 terabytes	media	hacked	3
329	329	Washington Post	2011	1270000	media	hacked	3

291 rows × 7 columns

```
sns.countplot(x='Organization type', data=df_org_upper);
plt.title('Number of data breaches per organization type')
plt.xticks(rotation=90);
```




```
sns.histplot(x='Organization type', stat='percent', data=df_org_upper,kde=True);
plt.title('Proportion of Data Breaches per Organization')
plt.xticks(rotation=90);
```



```
df_cleaned_records = df.drop(df[df.Records == 'unknown'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'G20 world leaders'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'tens of thousands'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '19 years of data'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '63 stores'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'over 5,000,000'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'unknown (client list)'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'millions'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '235 GB'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '350 clients emails'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '9,000,000 (approx) - basic booking, 2208 (credit card details)'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'Unknown'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '2.5GB'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '250 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '500 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '54 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '51 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '10 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '8 locations'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '93 stores'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '200 stores'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'undisclosed'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'Source Code Compromised'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == '100 terabytes'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'TBC'].index, inplace=True)
df_cleaned_records = df.drop(df[df.Records == 'unknown'].index, inplace=True)
df_cleaned_records = df.dropna(subset=['Records'])
```

```
df_cleaned_records.shape
```

```
(305, 6)
```

```
df_cleaned_records['Records'] = df_cleaned_records['Records'].astype(float)
```

```
<ipython-input-69-0438403b26b0>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df_cleaned_records['Records'] = df_cleaned_records['Records'].astype(float)
```

```
df_cleaned_records["Records"]
```

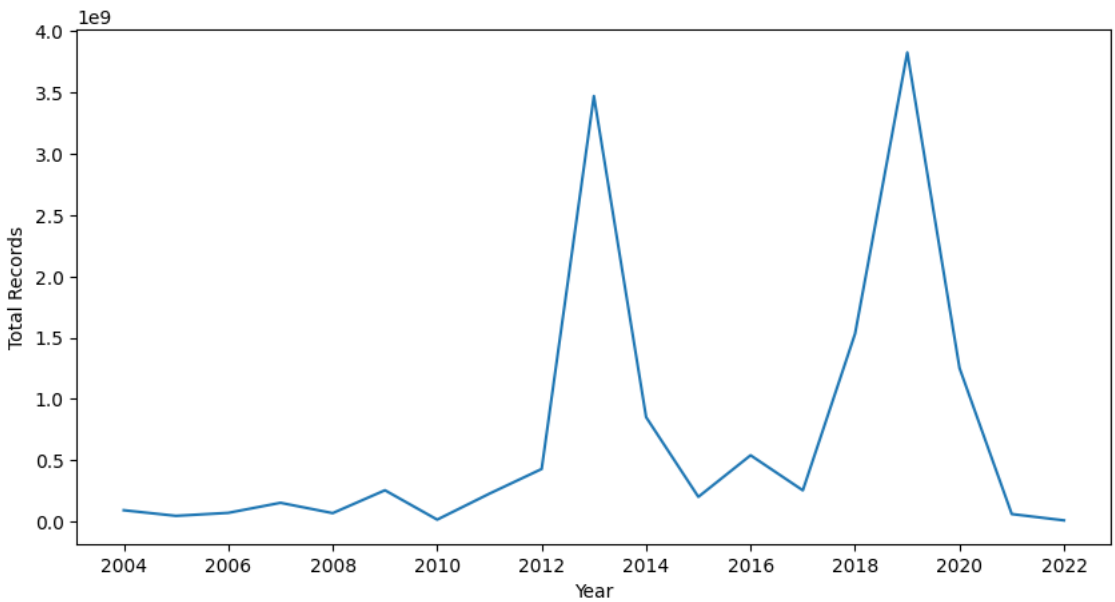
```
0        2200000.0
1       14870304.0
2         175350.0
3      152000000.0
4        7500000.0
          ...
347    173000000.0
348    200000000.0
349       391250.0
350      6400000.0
351        95000.0
Name: Records, Length: 305, dtype: float64
```

```
df_total_records = df_cleaned_records.groupby('Year', sort=False)["Records"].sum().reset_index(name ='Total Records')
```

```
df_total_records
```

	Year	Total Records
0	2016	5.405824e+08
1	2020	1.251422e+09
2	2013	3.469435e+09
3	2019	3.824901e+09
4	2017	2.547669e+08
5	2018	1.531850e+09
6	2005	4.682500e+07
7	2021	6.139627e+07
8	2015	2.016545e+08
9	2004	9.251000e+07
10	2006	7.126000e+07

```
plt.figure(figsize=(10,5))
sns.lineplot(data=df_total_records, x='Year', y='Total Records')
plt.xticks([2004, 2006, 2008, 2010, 2012, 2014, 2016, 2018, 2020, 2022]);
```



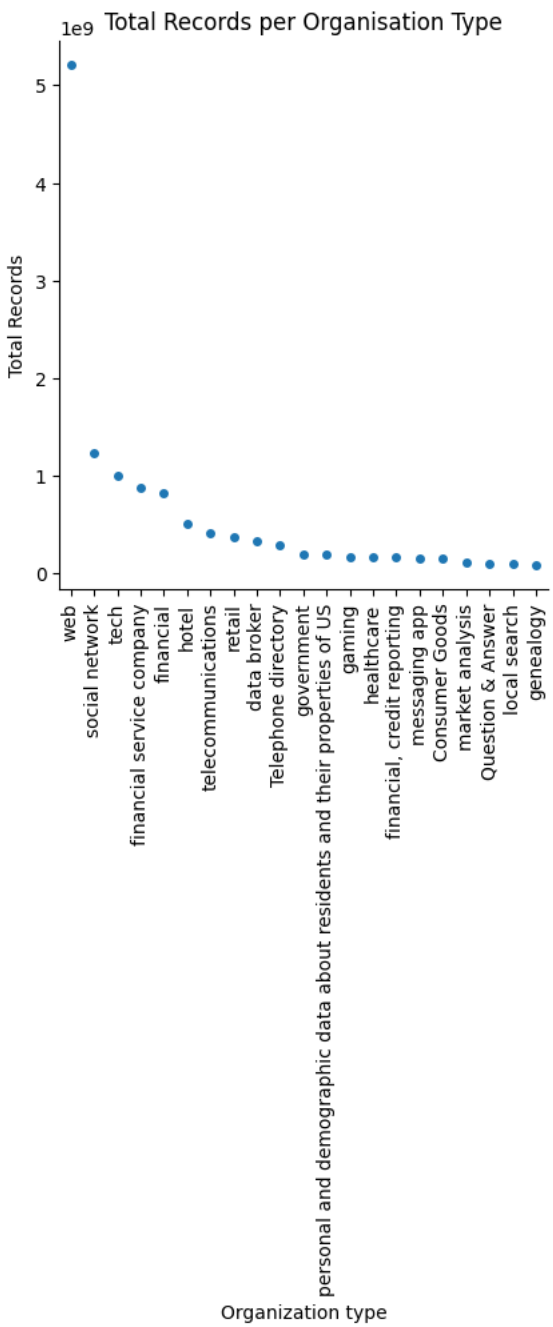

```
df_total_records_org = df_cleaned_records.groupby('Organization type', sort=False)["Records"].sum().reset_index(name ='Total Records')
```

```
df_total_records_org = df_total_records_org.sort_values('Total Records', ascending=False, ignore_index=True)
df_total_records_org_clean = df_total_records_org.drop(df_total_records_org.index[21:])
```

```
df_total_records_org_clean
```

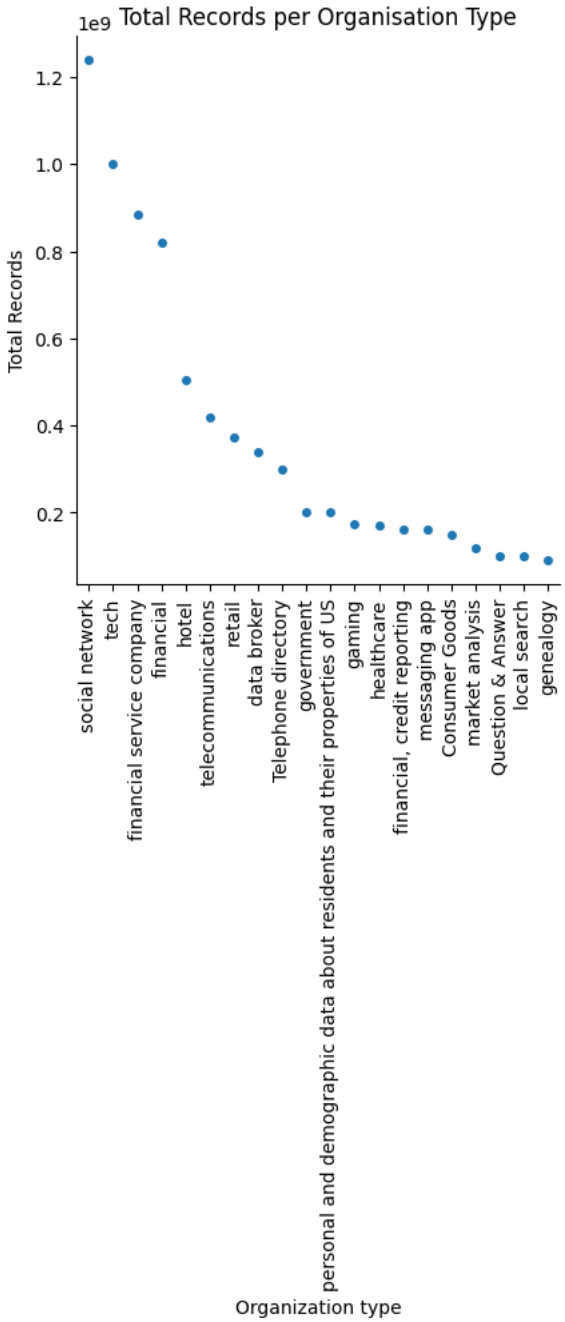
	Organization type	Total Records
0	web	5.203696e+09
1	social network	1.238000e+09
2	tech	1.000898e+09
3	financial service company	8.850000e+08
4	financial	8.185971e+08
5	hotel	5.055630e+08
6	telecommunications	4.200000e+08
7	retail	3.721407e+08
8	data broker	3.400000e+08
9	Telephone directory	2.990550e+08

```
sns.catplot(data=df_total_records_org_clean, x='Organization type', y='Total Records')
plt.title('Total Records per Organisation Type')
plt.xticks(rotation=90);
```



```
df_total_records_org_clean = df_total_records_org_clean.drop(df_total_records_org_clean.index[:1])
```

```
sns.catplot(data=df_total_records_org_clean, x='Organization type', y='Total Records')
plt.title('Total Records per Organisation Type')
plt.xticks(rotation=90);
```




```
df_total_records_Method = df_cleaned_records.groupby('Method', sort=False)["Records"].sum().reset_index(name ='Total Records')
```

```
df_total_records_Method = df_total_records_Method.sort_values('Total Records', ascending=False, ignore_index=True)
df_total_records_Method_clean = df_total_records_Method.drop(df_total_records_Method.index[21:])
df_total_records_Method_clean
```

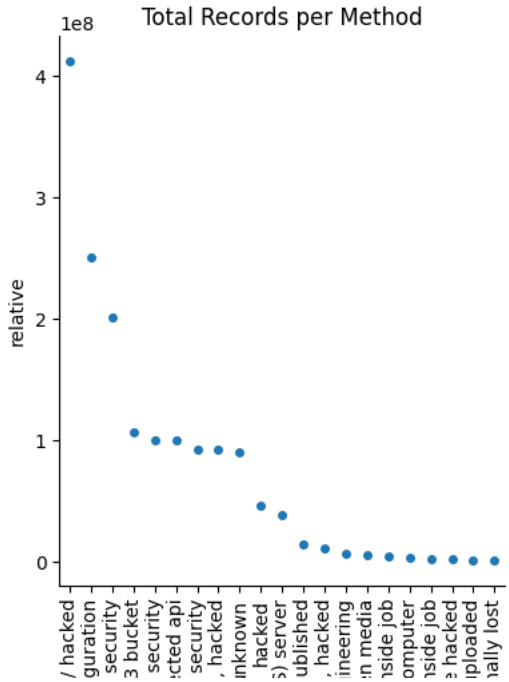
	Method	Total Records
0	hacked	7.404780e+09
1	poor security	3.610143e+09
2	unknown	4.482339e+08
3	poor security / hacked	4.122143e+08
4	accidentally published	2.699175e+08
5	data exposed by misconfiguration	2.500000e+08
6	Poor security	2.010000e+08
7	lost / stolen media	1.704345e+08
8	unsecured S3 bucket	1.060000e+08
9	unprotected api	1.000000e+08
10	misconfiguration/poor security	1.000000e+08
11	inside job, hacked	9.200000e+07
12	inside job	7.642610e+07
13	lost / stolen computer	4.139767e+07
14	publicly accessible Amazon Web Services (AWS) ...	3.800000e+07
15	improper setting, hacked	2.145775e+07

```
sns.catplot(data=df_total_records_Method_clean, x='Method', y='Total Records')
plt.title('Total Records per Method')
plt.xticks(rotation=90);
```


```
Method_counts = df['Method'].value_counts().rename('Method_counts')

df_Method = df_total_records_Method_clean.merge(Method_counts.to_frame(),
                                left_on='Method',
                                right_index=True)

df_Method['relative'] = df_Method['Total Records']/df_Method['Method_counts']

df_Method = df_Method.sort_values('relative', ascending=False, ignore_index=True)

df_Method
```

	Method	Total Records	Method_counts	relative
0	poor security / hacked	4.122143e+08	1	4.122143e+08
1	data exposed by misconfiguration	2.500000e+08	1	2.500000e+08
2	Poor security	2.010000e+08	1	2.010000e+08
3	unsecured S3 bucket	1.060000e+08	1	1.060000e+08
4	misconfiguration/poor security	1.000000e+08	1	1.000000e+08
5	unprotected api	1.000000e+08	1	1.000000e+08
6	poor security	3.610143e+09	39	9.256777e+07
7	inside job, hacked	9.200000e+07	1	9.200000e+07
8	unknown	4.482339e+08	5	8.964678e+07
9	hacked	7.404780e+09	160	4.627988e+07
10	publicly accessible Amazon Web Services (AWS) ...	3.800000e+07	1	3.800000e+07
11	accidentally published	2.699175e+08	19	1.420618e+07
12	improper setting, hacked	2.145775e+07	2	1.072888e+07
13	social engineering	6.054459e+06	1	6.054459e+06
14	lost / stolen media	1.704345e+08	32	5.326079e+06
15	inside job	7.642610e+07	18	4.245895e+06
16	lost / stolen computer	4.139767e+07	15	2.759844e+06
17	poor security/inside job	5.214200e+06	2	2.607100e+06
18	ransomware hacked	1.648922e+06	1	1.648922e+06
19	accidentally uploaded	1.500000e+06	1	1.500000e+06
20	intentionally lost	9.600000e+05	1	9.600000e+05

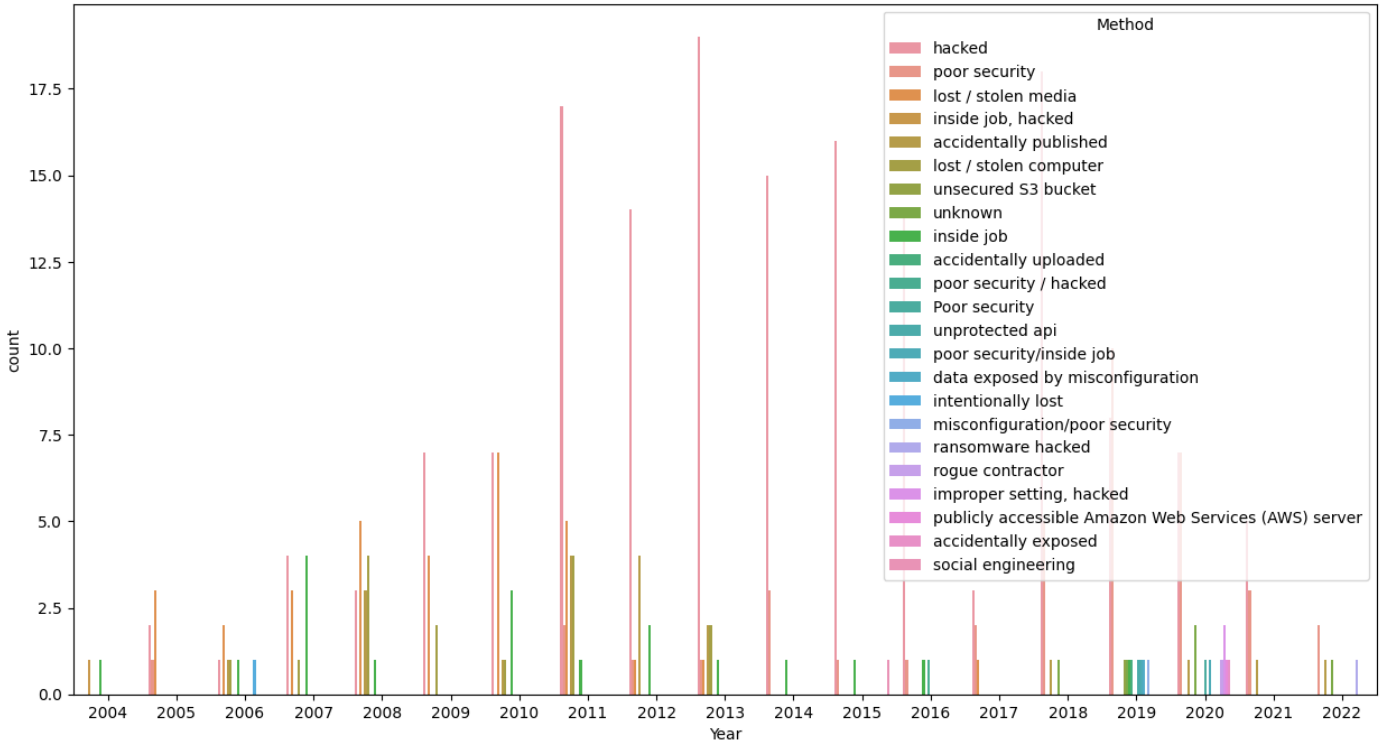
```
sns.catplot(data=df_Method, x='Method', y='relative')
plt.title('Total Records per Method')
plt.xticks(rotation=90);
```



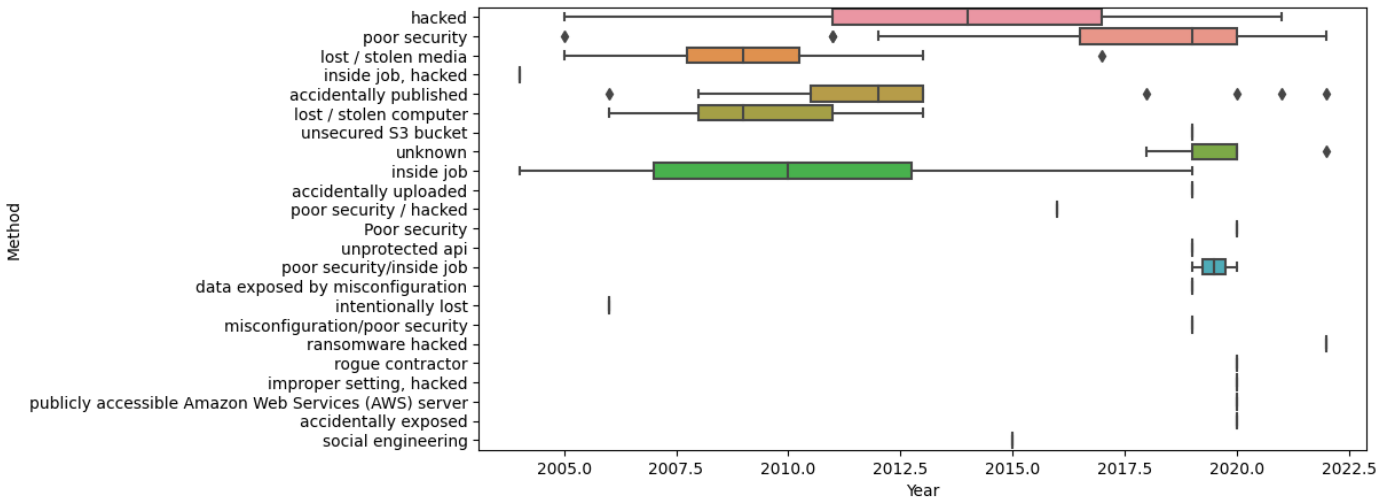

```
df.head()
```

	id	Entity	Year	Records	Organization type	Method
0	0	21st Century Oncology	2016	2200000	healthcare	hacked
1	1	500px	2020	14870304	social networking	hacked
2	2	Accendo Insurance Co.	2020	175350	healthcare	poor security
3	3	Adobe Systems Incorporated	2013	152000000	tech	hacked
4	4	Adobe Inc.	2019	7500000	tech	poor security

```
plt.figure(figsize=(15,8))
sns.countplot(data = df, x = 'Year', hue = 'Method');
```



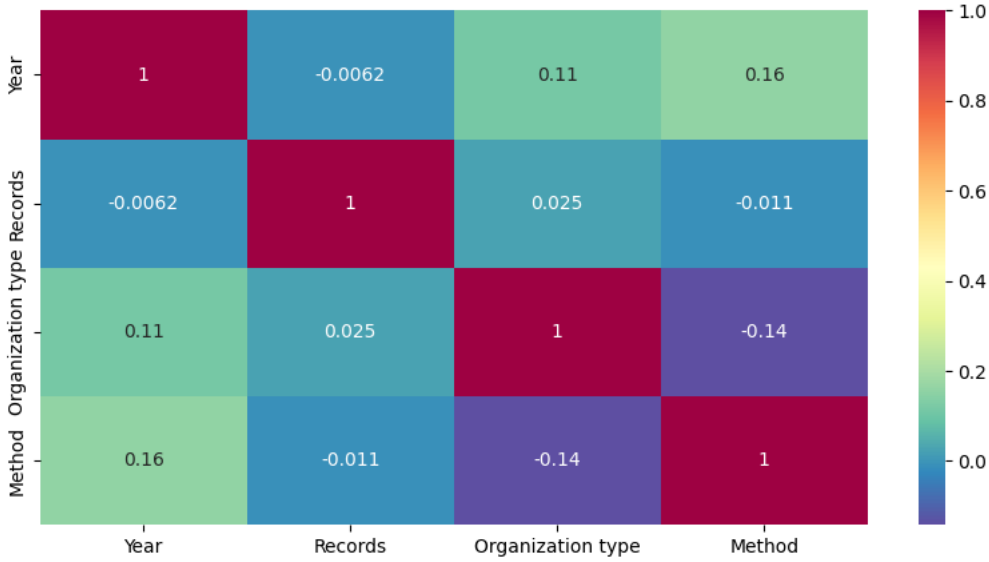
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y = 'Method', x = 'Year');
```




```
df_heatmap = df.copy(deep=True)
```

```
le = LabelEncoder()
df_heatmap['Records'] = le.fit_transform(df_heatmap['Records'])
df_heatmap['Entity'] = le.fit_transform(df_heatmap['Entity'])
df_heatmap['Organization type'] = le.fit_transform(df_heatmap['Organization type'])
df_heatmap['Method'] = le.fit_transform(df_heatmap['Method'])
```

```
plt.figure(figsize=(10,5))
sns.heatmap(df_heatmap[['Year', 'Records', 'Organization type', 'Method']].corr(), cmap='Spectral_r', annot=True);
```



Dataset 2

```
df2 = pd.read_csv("Cyber Security Breaches.csv")
```

```
df2.head()
```

duals_Affected	Date_of_Breach	Type_of_Breach	Location_of_Breached_Information	Date_Posted_or_Updated	Summary	breach_start	breacl
1000	10/16/2009	Theft	Paper	2014-06-30	A binder containing the protected health infor...	2009-10-16	
1000	9/22/2009	Theft	Network Server	2014-05-30	Five desktop computers containing unencrypted ...	2009-09-22	
501	10/12/2009	Theft	Other Portable Electronic Device, Other	2014-01-23	NaN	2009-10-12	
3800	10/9/2009	Loss	Laptop	2014-01-23	A laptop was lost by an employee while in tran...	2009-10-09	
					A shared Computer		

```
df2.columns = ['id', 'Number', 'Entity', 'State', 'Business_Associate_Involved', 'Individuals_Affected', 'Date_of_Breach',
               'Type_of_Breach', 'Location_of_Breached_Information', 'Date_Posted_or_Updated', 'Summary', 'breach_start',
               'breach_end', 'year']
```

```
df2.shape
```

```
(1055, 14)
```

```
df2.dtypes
```

```
id                                int64
Number                            int64
Entity                           object
State                            object
Business_Associate_Involved      object
Individuals_Affected              int64
Date_of_Breach                   object
Type_of_Breach                   object
Location_of_Breached_Information object
Date_Posted_or_Updated           object
Summary                          object
breach_start                     object
breach_end                       object
year                              int64
dtype: object
```

```
df2.isnull().sum()
```

```
id                                  0
Number                              0
Entity                              0
State                               0
Business_Associate_Involved       784
Individuals_Affected                0
Date_of_Breach                      0
Type_of_Breach                      0
Location_of_Breached_Information    0
Date_Posted_or_Updated              0
Summary                           913
breach_start                        0
breach_end                        910
year                                0
dtype: int64
```

```
df2.head()
```

	id	Number	Entity	State	Business_Associate_Involved	Individuals_Affected	Date_of_Breach	Type_of_Breach	Location_of_Breaches
0	1	0	Brooke Army Medical Center	TX	NaN	1000	10/16/2009	Theft	
1	2	1	Mid America Kidney Stone Association, LLC	MO	NaN	1000	9/22/2009	Theft	
2	3	2	Alaska Department of Health and Social Services	AK	NaN	501	10/12/2009	Theft	Other Portable Electroni
3	4	3	Health Services for Children with Special Need...	DC	NaN	3800	10/9/2009	Loss	
4	5	4	L. Douglas Carlson	CA	NaN	5257	9/27/2009	Theft	De

```
df2.drop(['Number','Summary', 'Date_Posted_or_Updated', 'breach_start',
          'breach_end', 'Business_Associate_Involved'], axis=1, inplace=True)
```

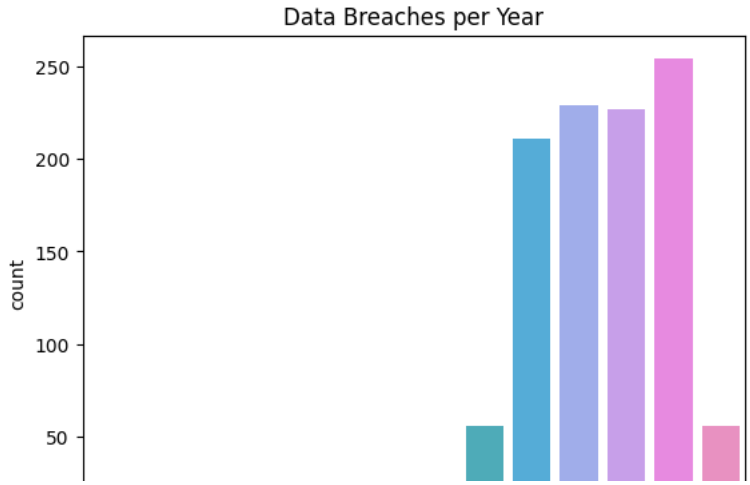
```
df2.head()
```

	id	Entity	State	Individuals_Affected	Date_of_Breach	Type_of_Breach	Location_of_Breached_Information	year
0	1	Brooke Army Medical Center	TX	1000	10/16/2009	Theft	Paper	2009
1	2	Mid America Kidney Stone Association, LLC	MO	1000	9/22/2009	Theft	Network Server	2009
2	3	Alaska Department of Health and Social Services	AK	501	10/12/2009	Theft	Other Portable Electronic Device, Other	2009
3	4	Health Services for Children with Special Need...	DC	3800	10/9/2009	Loss	Laptop	2009

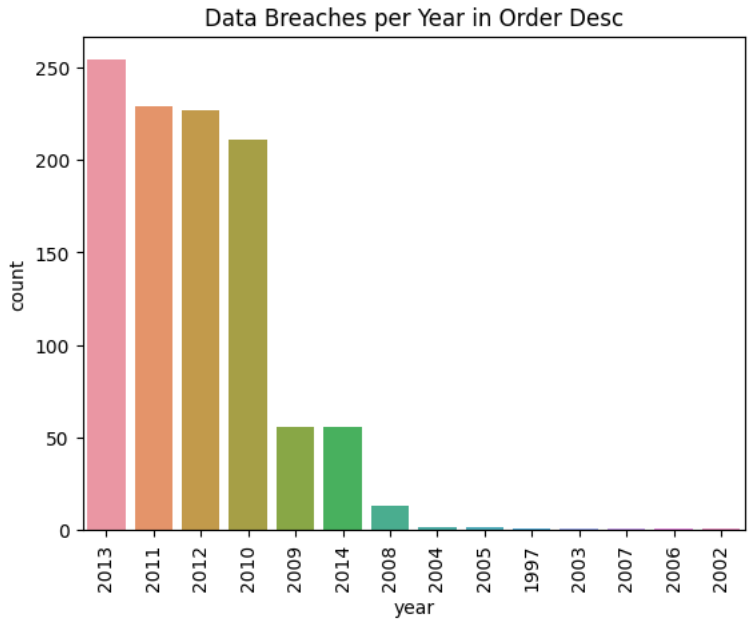
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1055 entries, 0 to 1054
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              1055 non-null   int64
 1   Entity                          1055 non-null   object
 2   State                           1055 non-null   object
 3   Individuals_Affected            1055 non-null   int64
 4   Date_of_Breach                  1055 non-null   object
 5   Type_of_Breach                  1055 non-null   object
 6   Location_of_Breached_Information 1055 non-null   object
 7   year                            1055 non-null   int64
dtypes: int64(3), object(5)
memory usage: 66.1+ KB
```

```
sns.countplot(data=df2, x='year');
plt.title('Data Breaches per Year')
plt.xticks(rotation=90);
```



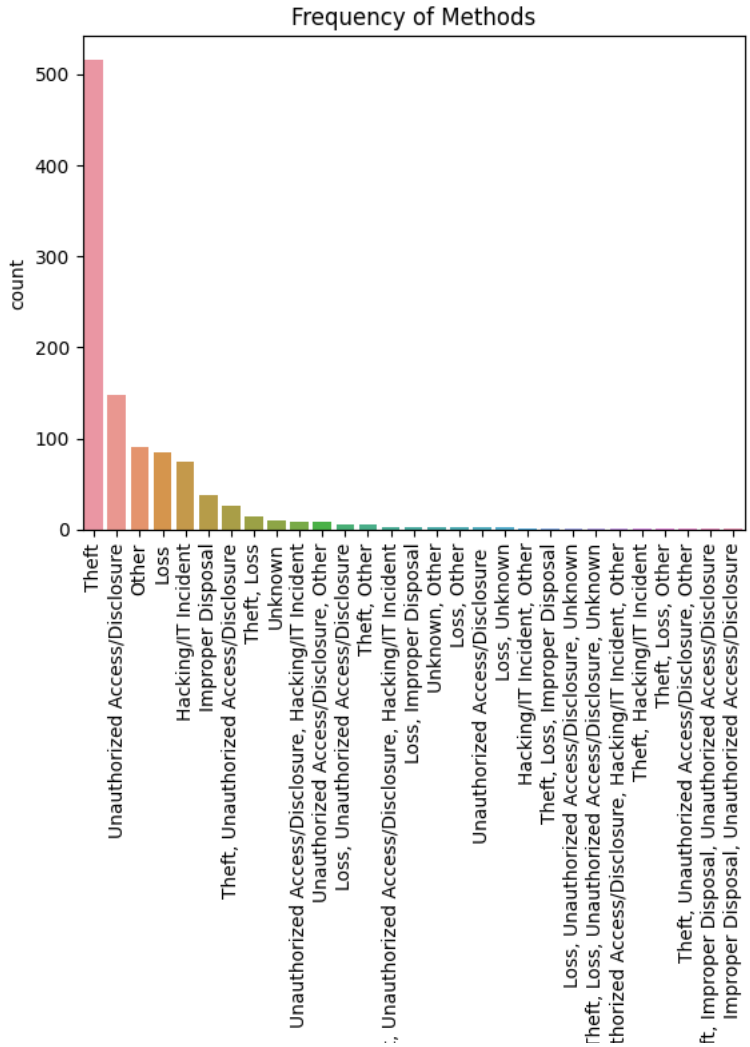

```
sns.countplot(data=df2, x='year', order = df2['year'].value_counts().index);
plt.title('Data Breaches per Year in Order Desc')
plt.xticks(rotation=90);
```




```
table_year_df2 = df2['year'].value_counts()
table_year_df2
```

```
2013    254
2011    229
2012    227
2010    211
2009     56
2014     56
2008     13
2004      2
2005      2
1997      1
2003      1
2007      1
2006      1
2002      1
Name: year, dtype: int64
```

```
sns.countplot(data=df2, x='Type_of_Breach', order = df2['Type_of_Breach'].value_counts().index);
plt.title('Frequency of Methods')
plt.xticks(rotation=90);
```




```
table3 = df2['Type_of_Breach'].value_counts()
table3.head(14)
```

```
Theft                                                          516
Unauthorized Access/Disclosure                                 148
Other                                                           91
Loss                                                            85
Hacking/IT Incident                                             75
Improper Disposal                                               38
Theft, Unauthorized Access/Disclosure                           26
Theft, Loss                                                     15
Unknown                                                         10
Unauthorized Access/Disclosure, Hacking/IT Incident              9
Unauthorized Access/Disclosure, Other                            8
Loss, Unauthorized Access/Disclosure                             5
Theft, Other                                                     5
Theft, Unauthorized Access/Disclosure, Hacking/IT Incident       3
Name: Type_of_Breach, dtype: int64
```

```
Type_of_Breach_counts = df2['Type_of_Breach'].value_counts().rename('Type_of_Breach_counts')

df2_Type_of_Breach = df2.merge(Type_of_Breach_counts.to_frame(),
                               left_on='Type_of_Breach',
                               right_index=True)
```

```
df2.head()
```

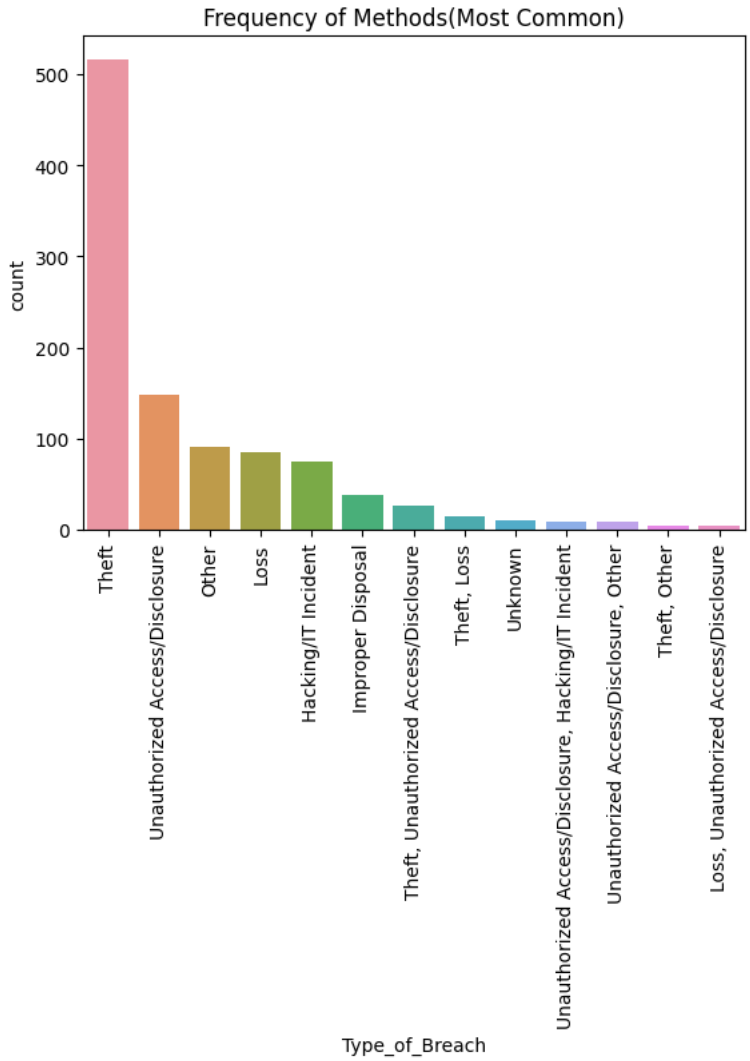
	id	Entity	State	Individuals_Affected	Date_of_Breach	Type_of_Breach	Location_of_Breached_Information	year
0	1	Brooke Army Medical Center	TX	1000	10/16/2009	Theft	Paper	2009
		Mid America Kidney Stone						
		Alaska Department of						

```
import plotly.express as px
```

```
df2_Type_of_Breach_upper = df2_Type_of_Breach[df2_Type_of_Breach.Type_of_Breach_counts > 4]
```

Need...

```
sns.countplot(data=df2_Type_of_Breach_upper, x='Type_of_Breach',
              order = df2_Type_of_Breach_upper['Type_of_Breach'].value_counts().index);
plt.title('Frequency of Methods(Most Common)')
plt.xticks(rotation=90);
```




```
table4 = df2['State'].value_counts()
table4
```

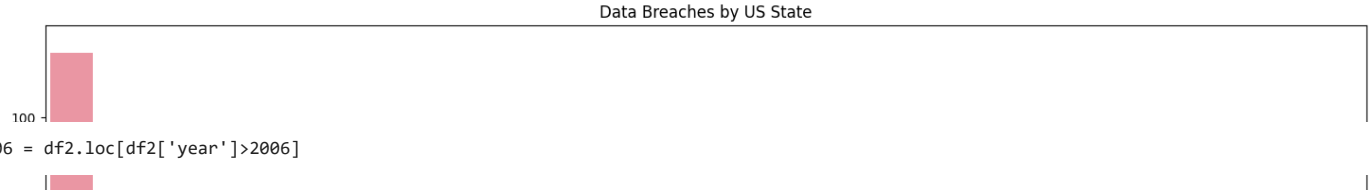
```
CA    113
TX     83
FL     66
NY     58
IL     49
PA     40
IN     40
OH     33
TN     32
NC     32
MA     32
PR     31
```

```
GA     30
KY     26
MI     26
MO     25
WA     25
AZ     21
MN     21
NJ     20
CO     18
VA     18
MD     18
CT     17
OR     15
WI     14
SC     13
AL     12
AR     11
NM     10
NE      9
UT      9
DC      9
IA      8
LA      7
RI      7
KS      7
OK      6
WV      5
MS      5
NV      5
AK      5
WY      4
NH      4
MT      4
DE      3
ND      3
ID      2
HI      1
SD      1
ME      1
VT      1
Name: State, dtype: int64
```

```
State_counts = df2['State'].value_counts().rename('State_counts')

df2_State = df2.merge(State_counts.to_frame(),
                              left_on='State',
                              right_index=True)
```

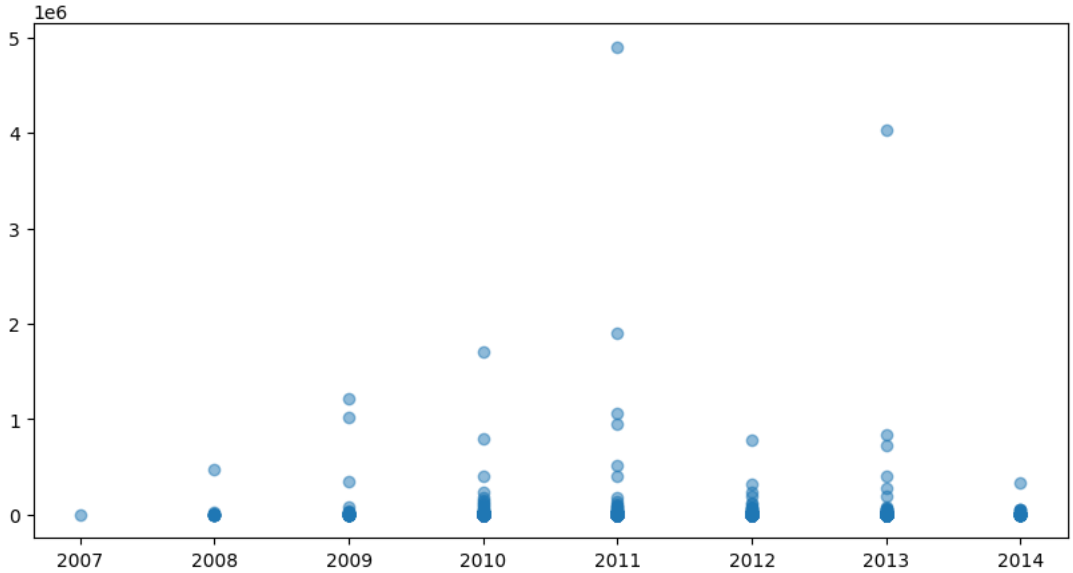
```
df2_State_upper = df2_State[df2_State.State_counts >= 15]
```

```
plt.figure(figsize=(18,8))
sns.countplot(data=df2_State_upper, x='State', order = df2_State_upper['State'].value_counts().index);
plt.title('Data Breaches by US State')
plt.xticks(rotation=90);
```

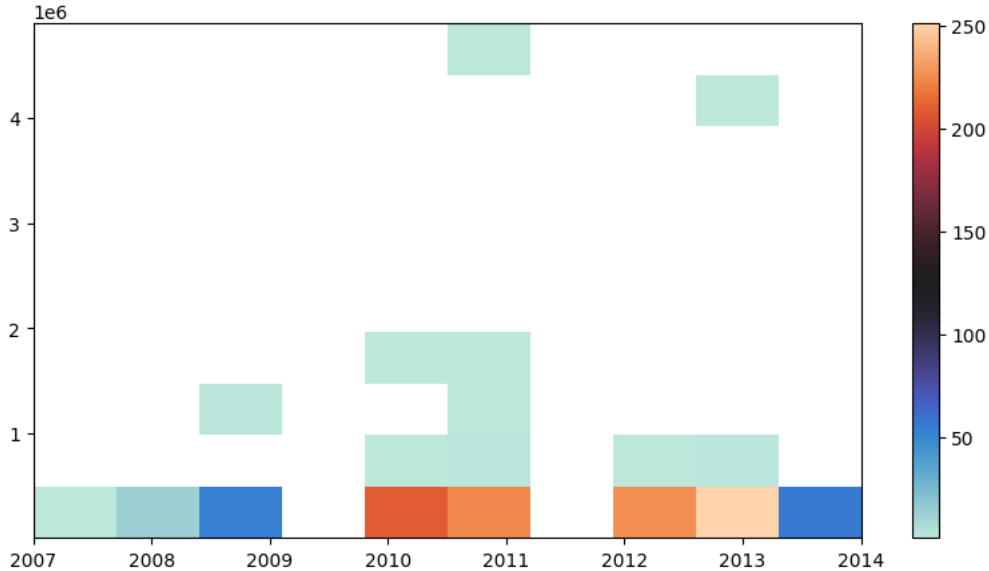



```
df2_2006 = df2.loc[df2['year']>2006]
```

```
plt.figure(figsize=(10,5))
plt.scatter(data = df2_2006, y = 'Individuals_Affected', x = 'year', alpha=1/2);
```

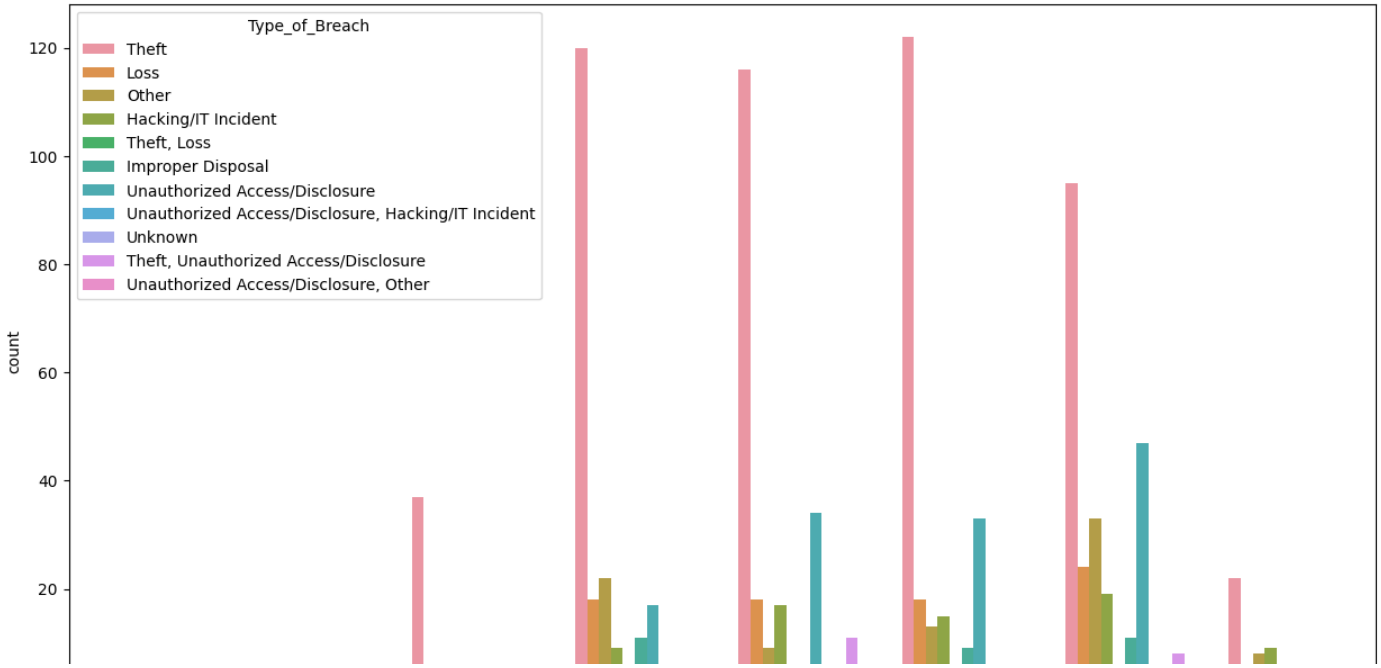



```
plt.figure(figsize=(10,5))
plt.hist2d(data = df2_2006,  y = 'Individuals_Affected', x = 'year', cmin=0.5, cmap = 'icefire')
plt.colorbar();
```

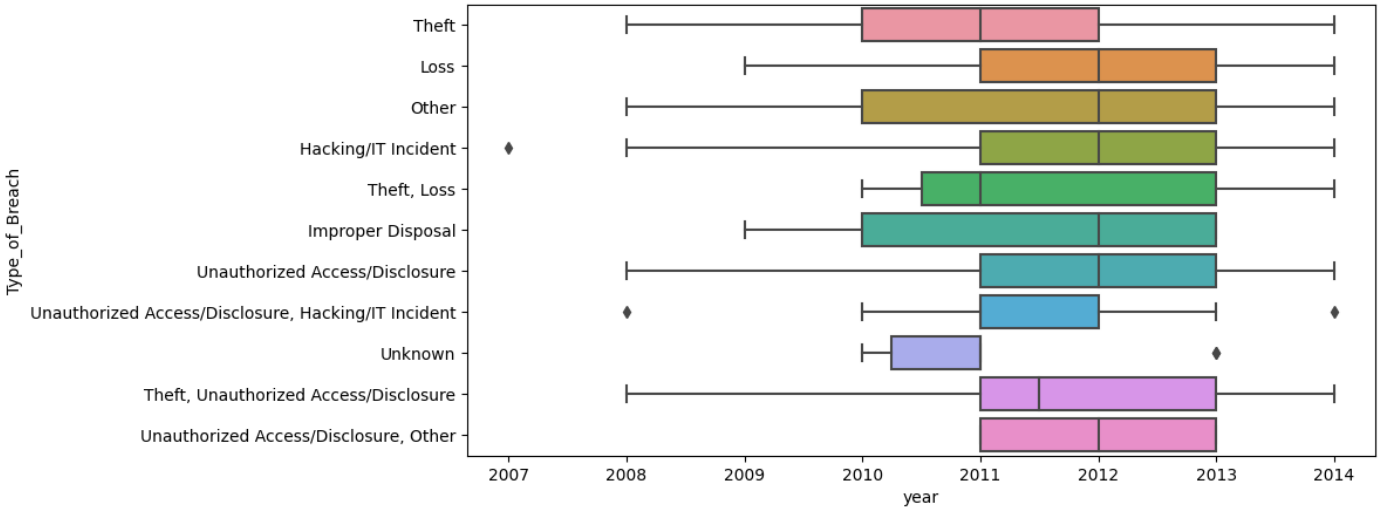



```
df2_2006_breach = df2_2006.loc[df2_2006['Type_of_Breach'].isin(df2_2006['Type_of_Breach'].value_counts().index[:11])]
```

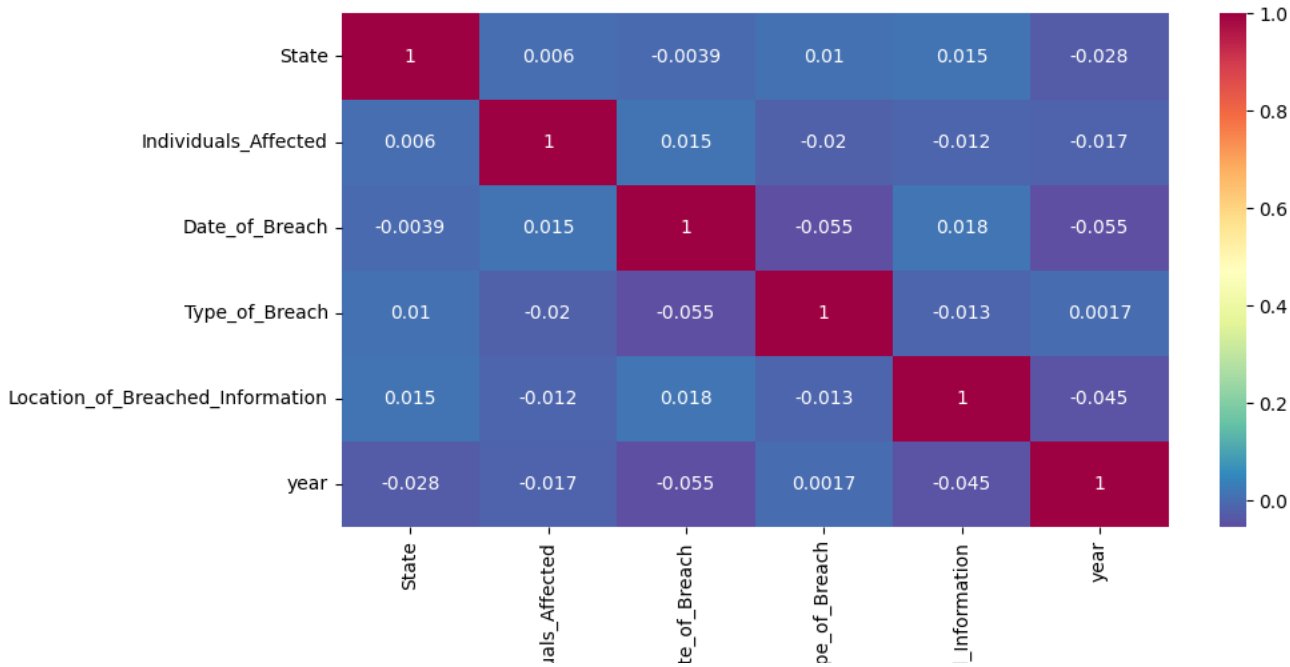
```
plt.figure(figsize=(15,8))
sns.countplot(data = df2_2006_breach, x = 'year', hue = 'Type_of_Breach');
```




```
plt.figure(figsize=(10,5))
sns.boxplot(data=df2_2006_breach, y = 'Type_of_Breach', x = 'year');
```




```
df2_heatmap = df2.copy(deep=True)


le = LabelEncoder()

df2_heatmap['State'] = le.fit_transform(df2_heatmap['State'])
df2_heatmap['Date_of_Breach'] = le.fit_transform(df2_heatmap['Date_of_Breach'])
df2_heatmap['Type_of_Breach'] = le.fit_transform(df2_heatmap['Type_of_Breach'])
df2_heatmap['Location_of_Breached_Information'] = le.fit_transform(df2_heatmap['Location_of_Breached_Information'])


plt.figure(figsize=(10,5))
sns.heatmap(df2_heatmap[['State', 'Individuals_Affected', 'Date_of_Breach', 'Type_of_Breach',
                 'Location_of_Breached_Information', 'year']].corr(), cmap='Spectral_r', annot=True);
```



Arima Code Starts

```
import statsmodels.api as sm
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307 entries, 0 to 351
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 307 non-null    int64
 1   Entity             307 non-null    object
 2   Year               307 non-null    int64
 3   Records            305 non-null    object
 4   Organization type  307 non-null    object
 5   Method             306 non-null    object
dtypes: int64(2), object(4)
memory usage: 16.8+ KB
```

```
# df2 = df_train.apply(pd.to_numeric,downcast="float")
df["Records"] = pd.to_numeric(df["Records"])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307 entries, 0 to 351
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 307 non-null    int64
 1   Entity             307 non-null    object
 2   Year               307 non-null    int64
 3   Records            305 non-null    float64
 4   Organization type  307 non-null    object
 5   Method             306 non-null    object
dtypes: float64(1), int64(2), object(3)
memory usage: 16.8+ KB
```

```
df.head()
```

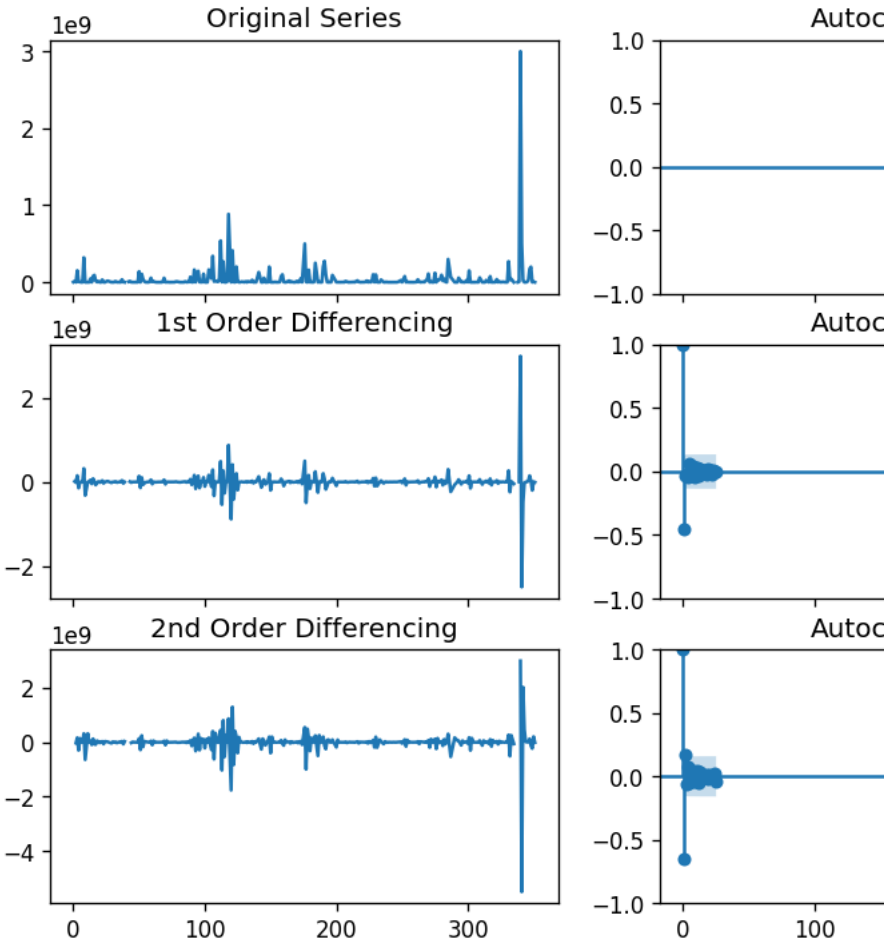
	id	Entity	Year	Records	Organization type	Method
0	0	21st Century Oncology	2016	2200000.0	healthcare	hacked
1	1	500px	2020	14870304.0	social networking	hacked

```
from statsmodels.tsa.stattools import adfuller
from numpy import log
result = adfuller(df.Year.dropna())
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

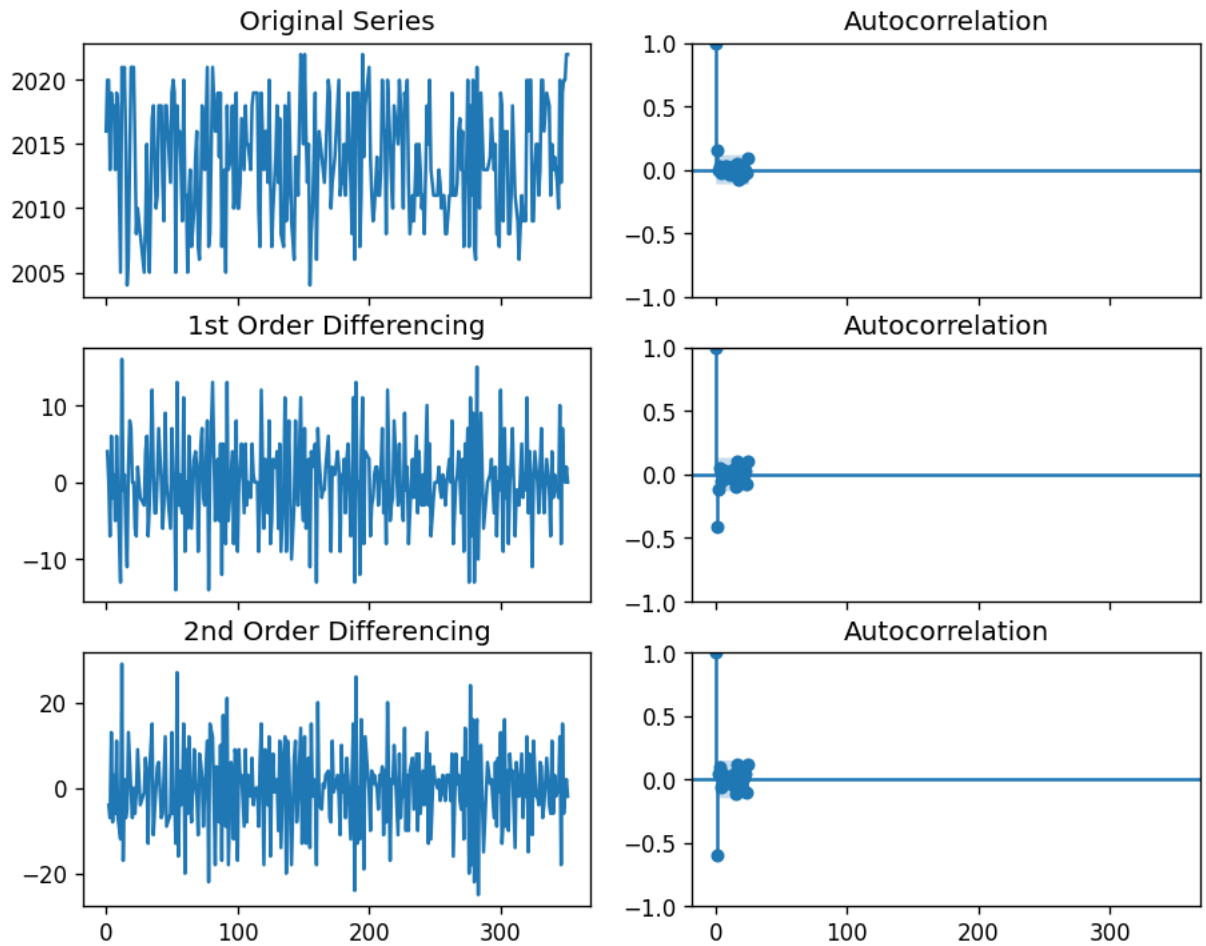
```
ADF Statistic: -14.774299
p-value: 0.000000
```

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})


# Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df.Records); axes[0, 0].set_title('Original Series')
plot_acf(df.Records, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(df.Records.diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df.Records.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(df.Records.diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(df.Records.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```

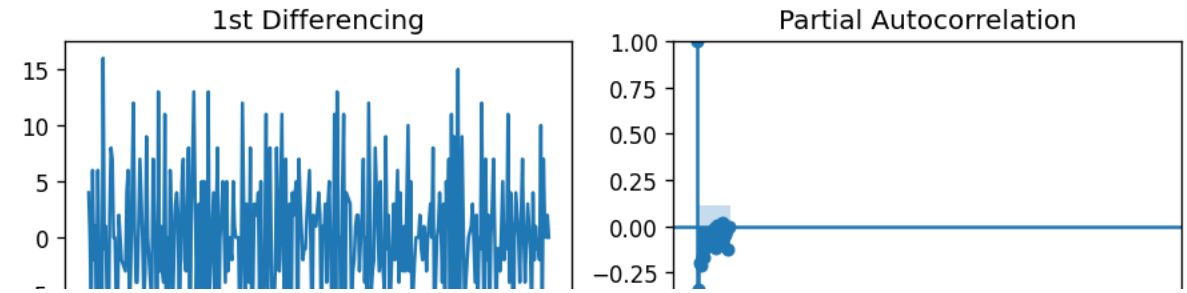
```
/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_base.py:2503: UserWarning:

Warning: converting a masked element to nan.
```



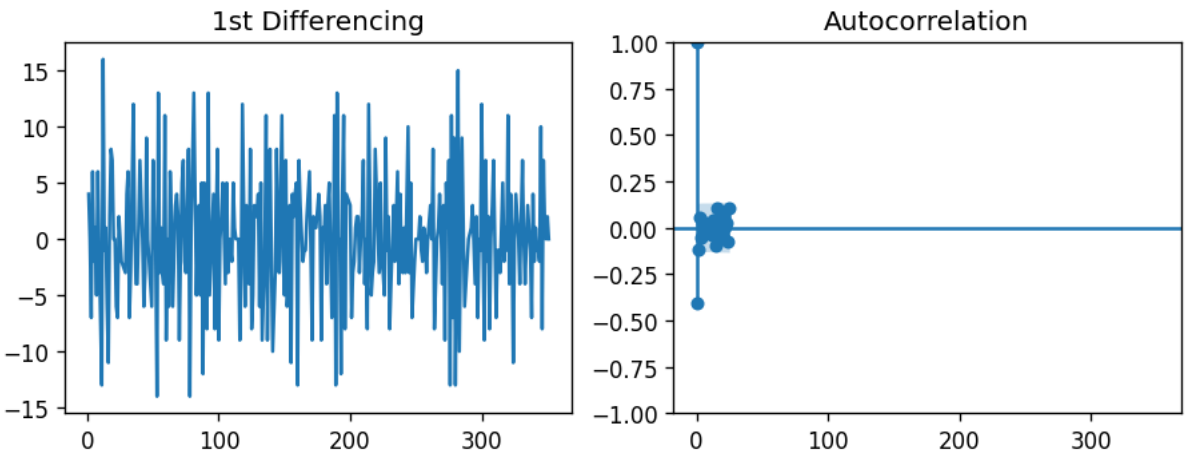
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})


# Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df.Year); axes[0, 0].set_title('Original Series')
plot_acf(df.Year, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(df.Year.diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df.Year.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(df.Year.diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(df.Year.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```




```
plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(df.Year.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,5))
plot_pacf(df.Year.diff().dropna(), ax=axes[1])

plt.show()
```




```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(df.Year.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,1.2))
plot_acf(df.Year.diff().dropna(), ax=axes[1])

plt.show()
```




```
from statsmodels.tsa.arima_model import ARIMA

model = sm.tsa.arima.ARIMA(df["Records"].value_count().index, order=(1,1,2))
result = model.fit()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-190-7a2e6248e4c8> in <cell line: 3>()
      1 from statsmodels.tsa.arima_model import ARIMA
      2
----> 3 model = sm.tsa.arima.ARIMA(df["Records"].value_count().index, order=(1,1,2))
      4 result = model.fit()

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   5900         ):
   5901             return self[name]
-> 5902         return object.__getattribute__(self, name)
   5903
   5904     def __setattr__(self, name: str, value) -> None:

AttributeError: 'Series' object has no attribute 'value_count'
```

SEARCH STACK OVERFLOW

```
df.head()
```

	id	Entity	Year	Records	Organization type	Method
0	0	21st Century Oncology	2016	2200000.0	healthcare	hacked
1	1	500px	2020	14870304.0	social networking	hacked
2	2	Accendo Insurance Co.	2020	175350.0	healthcare	poor security
3	3	Adobe Systems Incorporated	2013	152000000.0	tech	hacked