Fatalites I&P

October 22, 2023

1 Comprehensive Analysis of Fatalities in the Israeli Palestinian Conflict: Demographics, Causes, and Geographical Patterns

[1]:
```
# Project By: Uvesh Ahmad
# Data Set Link: https://github.com/Uvesh-Ahmad
# Portfolio: https://uvesh-ahmad.github.io/uvesh.ah/
```

[2]:
```
from IPython.display import Image
Image(filename='C:\\Users\\mrala\\OneDrive\\Desktop\\Python Data\\Data Analyst␣
↪project\\Both Analyst projectIsrael-Palestine Civilsdaily.jpg')
```

[2]:

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

[4]:
```
import warnings
warnings.filterwarnings('ignore')
```

[5]:
```
df = pd.read_csv('C:\\Users\\mrala\\OneDrive\\Desktop\\Python Data\\Data␣
↪Analyst project\\fatalities_isr_pse_conflict_2000_to_2023.csv')
```

[6]:
```
df.head()
```

[6]:
```
                                          name date_of_event   age  \
0  'Abd a-Rahman Suleiman Muhammad Abu Daghash    2023-09-24  32.0
1         Usayed Farhan Muhammad 'Ali Abu 'Ali    2023-09-24  21.0
2             'Abdallah 'Imad Sa'ed Abu Hassan    2023-09-22  16.0
3             Durgham Muhammad Yihya al-Akhras    2023-09-20  19.0
4                 Raafat 'Omar Ahmad Khamaisah    2023-09-19  15.0

   citizenship     event_location event_location_district  \
0  Palestinian     Nur Shams R.C.                 Tulkarm
1  Palestinian     Nur Shams R.C.                 Tulkarm
2  Palestinian           Kfar Dan                   Jenin
3  Palestinian  'Aqbat Jaber R.C.                 Jericho
4  Palestinian         Jenin R.C.                   Jenin

  event_location_region date_of_death gender took_part_in_the_hostilities  \
0             West Bank    2023-09-24      M                          NaN
1             West Bank    2023-09-24      M                          NaN
2             West Bank    2023-09-22      M                          NaN
3             West Bank    2023-09-20      M                          NaN
4             West Bank    2023-09-19      M                          NaN

  place_of_residence place_of_residence_district type_of_injury  \
0     Nur Shams R.C.                     Tulkarm        gunfire
1     Nur Shams R.C.                     Tulkarm        gunfire
2           al-Yamun                       Jenin        gunfire
3  'Aqbat Jaber R.C.                     Jericho        gunfire
4              Jenin                       Jenin        gunfire

       ammunition               killed_by  \
0  live ammunition  Israeli security forces
1  live ammunition  Israeli security forces
2  live ammunition  Israeli security forces
3  live ammunition  Israeli security forces
4  live ammunition  Israeli security forces

                                          notes
0  Fatally shot by Israeli forces while standing …
1  Fatally shot by Israeli forces while trying to…
2  Fatally shot by soldiers while firing at them …
3  Shot in the head by Israeli forces while throw…
4  Wounded by soldiers' gunfire after running awa…
```

[7]:
```
df.tail()
```

[7]:
```
                            name date_of_event   age  citizenship  \
11119            Binyamin Herling    2000-10-19  64.0      Israeli
11120  Farid Musa 'Issa a-Nesasreh    2000-10-17  28.0  Palestinian
11121            Hillel Lieberman    2000-10-07  36.0      Israeli
11122   Fahed Mustafa 'Odeh Baker    2000-10-07  21.0  Palestinian
11123           Wichlav Zalsevsky    2000-10-02  24.0      Israeli

      event_location event_location_district event_location_region  \
11119         Nablus                  Nablus             West Bank
11120     Beit Furik                  Nablus             West Bank
11121         Nablus                  Nablus             West Bank
11122          Bidya                  Salfit             West Bank
11123          Masha                  Salfit             West Bank

      date_of_death gender took_part_in_the_hostilities place_of_residence  \
11119    2000-10-19      M                    Israelis            Kedumim
11120    2000-10-17      M                     Unknown         Beit Furik
11121    2000-10-07      M                    Israelis         Elon Moreh
11122    2000-10-07      M                          No              Bidya
11123    2000-10-02      M                    Israelis             Ashdod

      place_of_residence_district type_of_injury       ammunition  \
11119                    Tulkarm        gunfire  live ammunition
11120                     Nablus        gunfire              NaN
11121                     Nablus        gunfire  live ammunition
11122                     Salfit        gunfire              NaN
11123                     Israel        gunfire  live ammunition

                killed_by  \
11119  Palestinian civilians
11120      Israeli civilians
11121  Palestinian civilians
11122      Israeli civilians
11123  Palestinian civilians

                                            notes
11119                 Killed while hiking on Mt. Eival.
11120  Killed by a settler from Itamar while harvesti…
11121    His body was found a day after he disappeared.
11122   Killed by settlers who rioted in Biddya village.
11123                                             NaN
```

[8]:
```
df.shape
```

[8]:
```
(11124, 16)
```

[9]:
```
df.columns
```

[9]:
```
Index(['name', 'date_of_event', 'age', 'citizenship', 'event_location',
       'event_location_district', 'event_location_region', 'date_of_death',
       'gender', 'took_part_in_the_hostilities', 'place_of_residence',
       'place_of_residence_district', 'type_of_injury', 'ammunition',
       'killed_by', 'notes'],
      dtype='object')
```

[10]:
```
df.duplicated().sum()
```

[10]:
```
7
```

[11]:
```
df = df.drop_duplicates()
```

[12]:
```
df.isnull().sum()
```

[12]:
```
name                              0
date_of_event                     0
age                             122
citizenship                       0
event_location                    0
event_location_district           0
event_location_region             0
date_of_death                     0
gender                           14
took_part_in_the_hostilities   1430
place_of_residence               61
place_of_residence_district      61
type_of_injury                  290
ammunition                     5246
killed_by                         0
notes                           277
dtype: int64
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11117 entries, 0 to 11123
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   name                          11117 non-null  object
 1   date_of_event                 11117 non-null  object
 2   age                           10995 non-null  float64
 3   citizenship                   11117 non-null  object
 4   event_location                11117 non-null  object
 5   event_location_district       11117 non-null  object
 6   event_location_region         11117 non-null  object
 7   date_of_death                 11117 non-null  object
```

```
 8   gender                          11103 non-null  object
 9   took_part_in_the_hostilities    9687 non-null   object
 10  place_of_residence              11056 non-null  object
 11  place_of_residence_district     11056 non-null  object
 12  type_of_injury                  10827 non-null  object
 13  ammunition                      5871 non-null   object
 14  killed_by                       11117 non-null  object
 15  notes                           10840 non-null  object
dtypes: float64(1), object(15)
memory usage: 1.4+ MB
```

[14]:
```
df.describe()
```

[14]:
```
                age
count  10995.000000
mean      26.745703
std       13.780548
min        1.000000
25%       19.000000
50%       23.000000
75%       31.000000
max      112.000000
```

[15]:
```
from numpy import true_divide


df['age'].fillna(df['age'].mean(), inplace=True)
df['gender'].fillna('Unknown', inplace=True)
df['took_part_in_the_hostilities'].fillna('Not Specified', inplace=True)
df['place_of_residence'].fillna('Unknown', inplace=True)
df['place_of_residence_district'].fillna('Unknown', inplace=True)
df['type_of_injury'].fillna('Unknown', inplace=True)
df['ammunition'].fillna('Unknown', inplace=True)
df['notes'].fillna('Unknown', inplace=True)
```

[16]:
```
df.isnull().sum()
```

[16]:
```
name                            0
date_of_event                   0
age                             0
citizenship                     0
event_location                  0
event_location_district         0
event_location_region           0
date_of_death                   0
gender                          0
took_part_in_the_hostilities    0
place_of_residence              0
```

```
place_of_residence_district     0
type_of_injury                  0
ammunition                      0
killed_by                       0
notes                           0
dtype: int64
```

[17]:
```
df['date_of_event'] = pd.to_datetime(df['date_of_event'])
df['date_of_death'] = pd.to_datetime(df['date_of_death'])
```

[18]:
```
df.nunique()
```

[18]:
```
name                           11083
date_of_event                   2405
age                               96
citizenship                        4
event_location                   494
event_location_district           20
event_location_region              3
date_of_death                   2593
gender                             3
took_part_in_the_hostilities       6
place_of_residence               581
place_of_residence_district       21
type_of_injury                    14
ammunition                        22
killed_by                          3
notes                           6745
dtype: int64
```

[19]:
```
object_columns = df.select_dtypes(include=[object]).columns
print("Object type columns:")
print(object_columns)


numrical_columns = df.select_dtypes(include=['int', 'float']).columns
print("\nNumerical type columns:")
print(numrical_columns)
```

```
Object type columns:
Index(['name', 'citizenship', 'event_location', 'event_location_district',
       'event_location_region', 'gender', 'took_part_in_the_hostilities',
       'place_of_residence', 'place_of_residence_district', 'type_of_injury',
       'ammunition', 'killed_by', 'notes'],
      dtype='object')

Numerical type columns:
```

```
Index(['age'], dtype='object')
```

[20]:
```python
import numpy as np

def classify_features(df):
    categorical_features = []
    non_categorical_features = []
    discrete_features = []
    continuous_features = []

    for column in df.columns:
        if df[column].dtype == 'object':
            if df[column].nunique() < 10:
                categorical_features.append(column)
            else:
                non_categorical_features.append(column)
        elif np.issubdtype(df[column].dtype, np.integer) or np.
 ↪issubdtype(df[column].dtype, np.floating):
            if df[column].nunique() < 10:
                discrete_features.append(column)
            else:
                continuous_features.append(column)

    return categorical_features, non_categorical_features, discrete_features,␣
 ↪continuous_features

# Example usage
# Assuming 'data' is your DataFrame
# categorical, non_categorical, discrete, continuous = classify_features(data)
```

[21]:
```python
categorical, non_categorical, discrete, continuous = classify_features(df)
```

[22]:
```python
from pyexpat import features


print("Categorical_Features:", categorical )
print("Non-Categorical Features: ", non_categorical)
print("Discrete_Features:", discrete )
print("Continuous Features:", continuous)
```

```
Categorical_Features: ['citizenship', 'event_location_region', 'gender',
'took_part_in_the_hostilities', 'killed_by']
Non-Categorical Features:  ['name', 'event_location', 'event_location_district',
'place_of_residence', 'place_of_residence_district', 'type_of_injury',
'ammunition', 'notes']
Discrete_Features: []
Continuous Features: ['age']
```

[23]:
```
for i in categorical:
    print(i, ':')
    print(df[i].unique())
    print('\n')
```

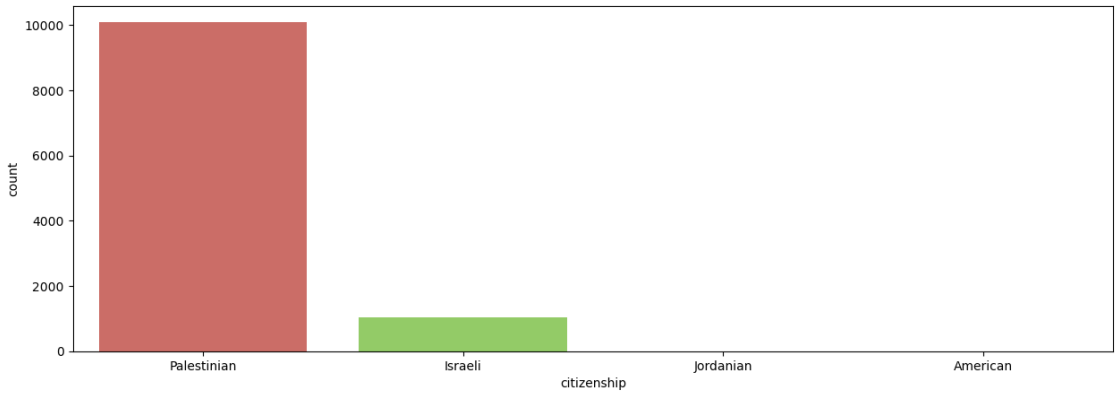
```
citizenship :
['Palestinian' 'Israeli' 'Jordanian' 'American']


event_location_region :
['West Bank' 'Gaza Strip' 'Israel']


gender :
['M' 'F' 'Unknown']


took_part_in_the_hostilities :
['Not Specified' 'No' 'Yes' 'Unknown' 'Israelis'
 'Object of targeted killing']


killed_by :
['Israeli security forces' 'Palestinian civilians' 'Israeli civilians']
```

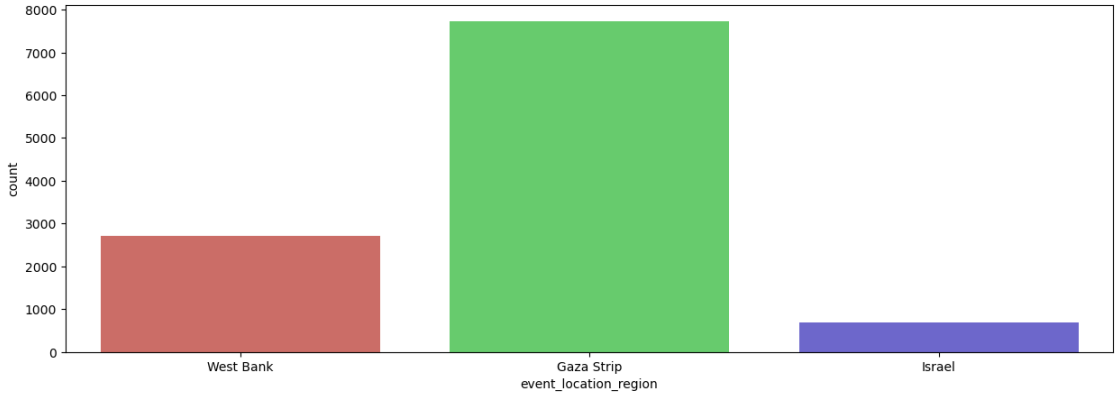
[24]:
```
for i in categorical:
    print(i,':')
    print(df[i].value_counts())
    print('\n')
```

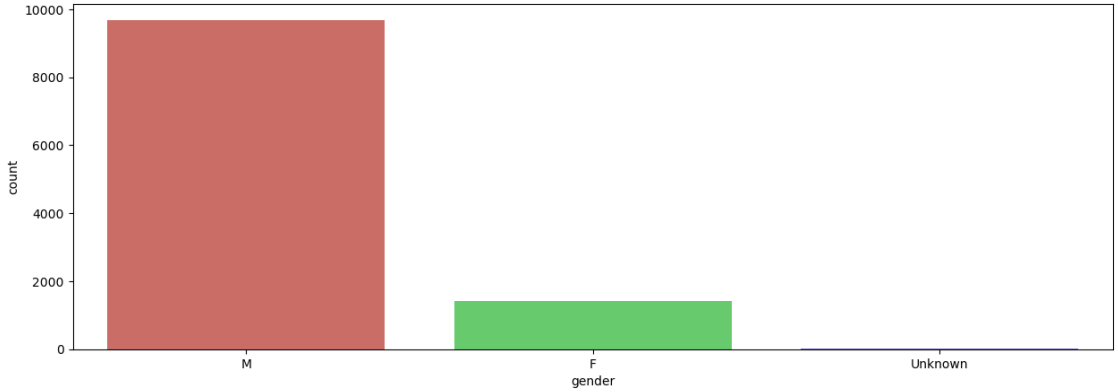
```
citizenship :
Palestinian    10085
Israeli         1029
Jordanian          2
American           1
Name: citizenship, dtype: int64


event_location_region :
Gaza Strip    7731
West Bank     2708
Israel         678
Name: event_location_region, dtype: int64


gender :
```

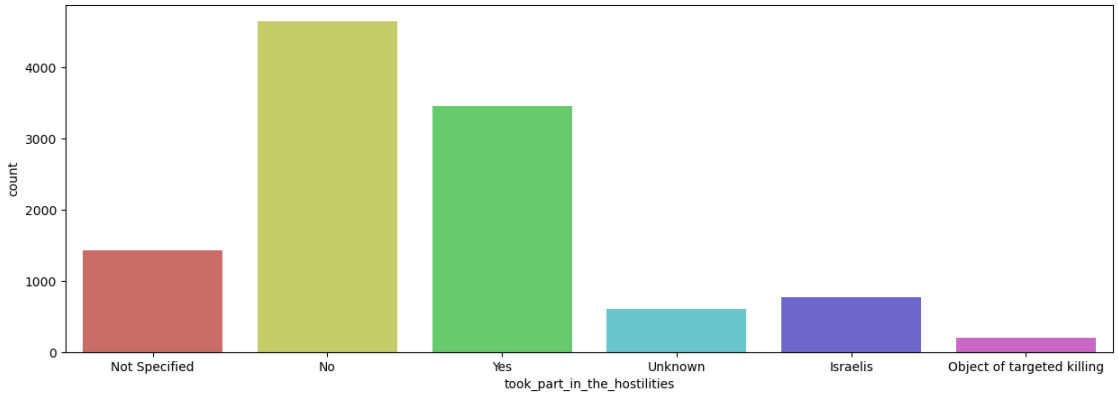
```
M          9680
F          1423
Unknown      14
Name: gender, dtype: int64


took_part_in_the_hostilities :
No                            4653
Yes                           3465
Not Specified                 1430
Israelis                       771
Unknown                        598
Object of targeted killing     200
Name: took_part_in_the_hostilities, dtype: int64


killed_by :
Israeli security forces    9993
Palestinian civilians      1028
Israeli civilians            96
Name: killed_by, dtype: int64
```

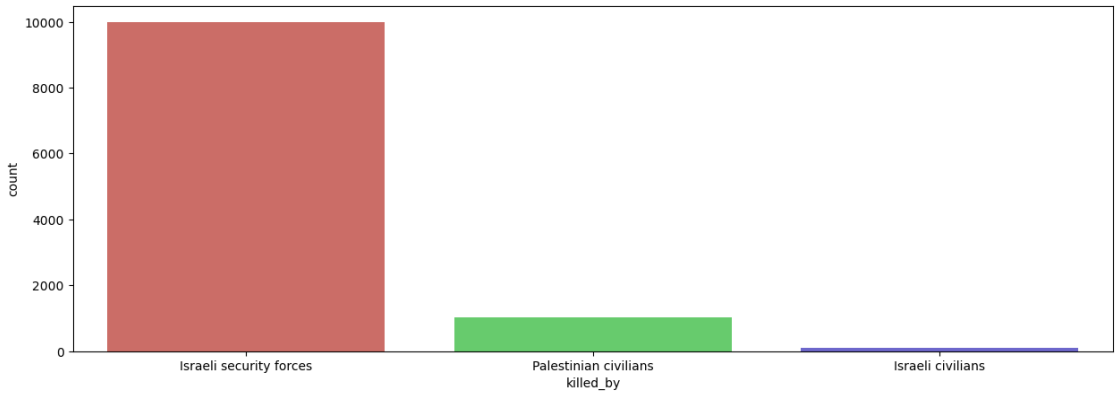
[25]:
```
for i in categorical:
    plt.figure(figsize=(15,5))
    sns.countplot(x=i, data=df, palette='hls')
    plt.show()
```



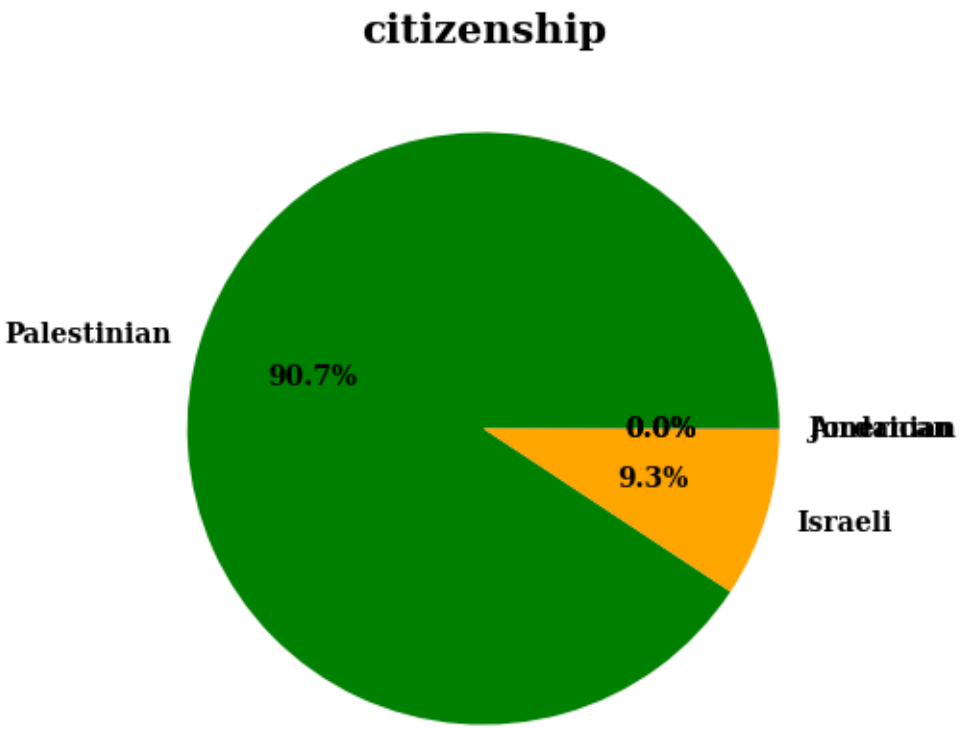

8000
7000
6000
5000
4000
3000
2000
1000
0
count
West Bank
Gaza Strip
Israel
event_location_region

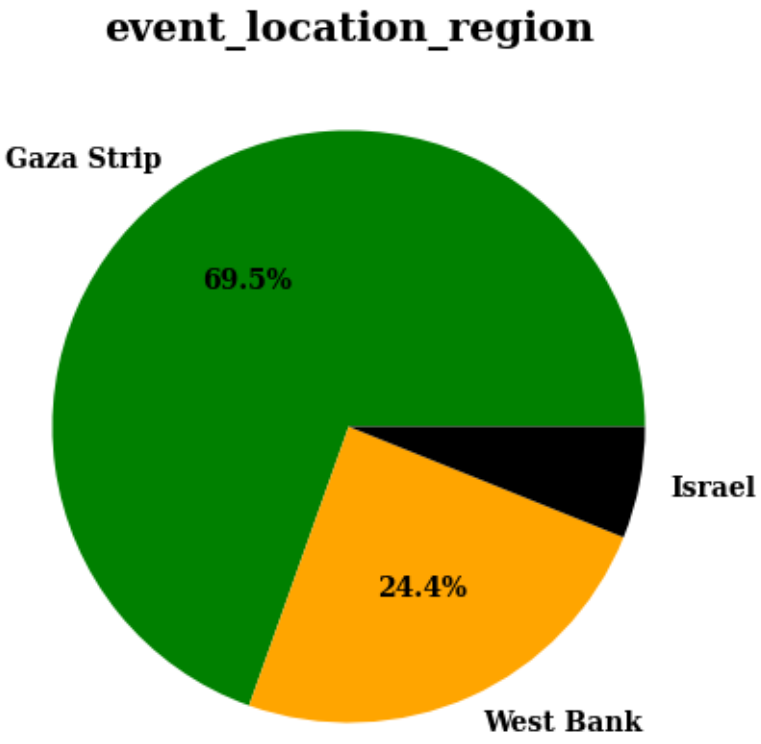

10000
8000
6000
4000
2000
0
count
M
F
Unknown
gender

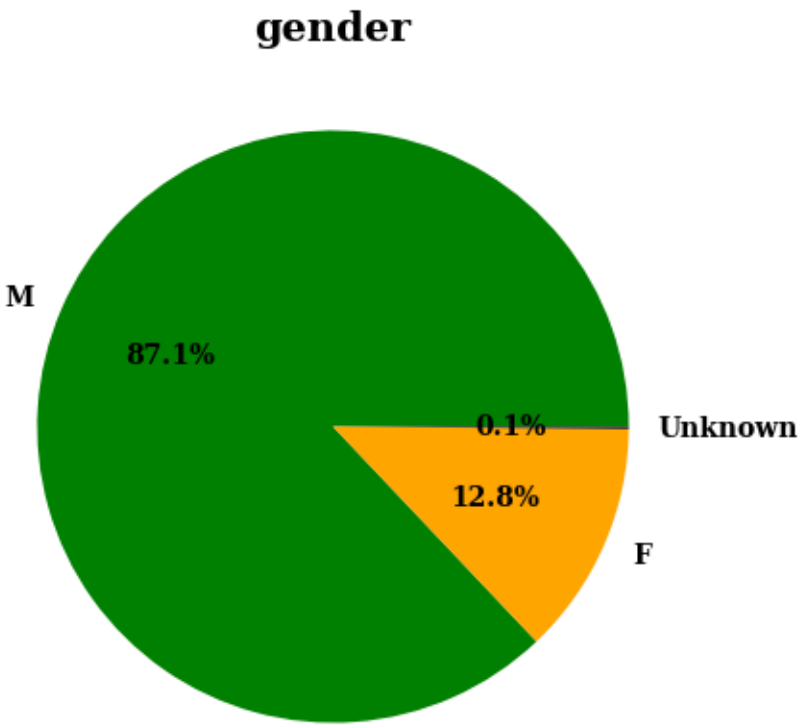

4000
3000
2000
1000
0
count
Not Specified
No
Yes
Unknown
Israelis
Object of targeted killing
took_part_in_the_hostilities

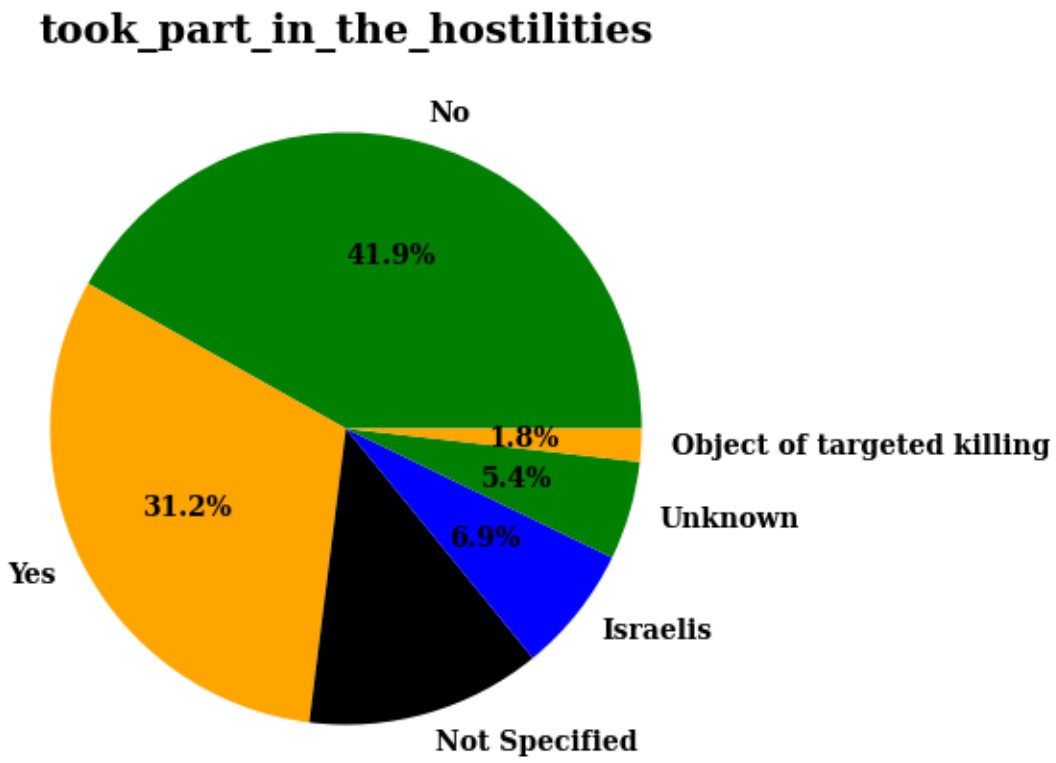


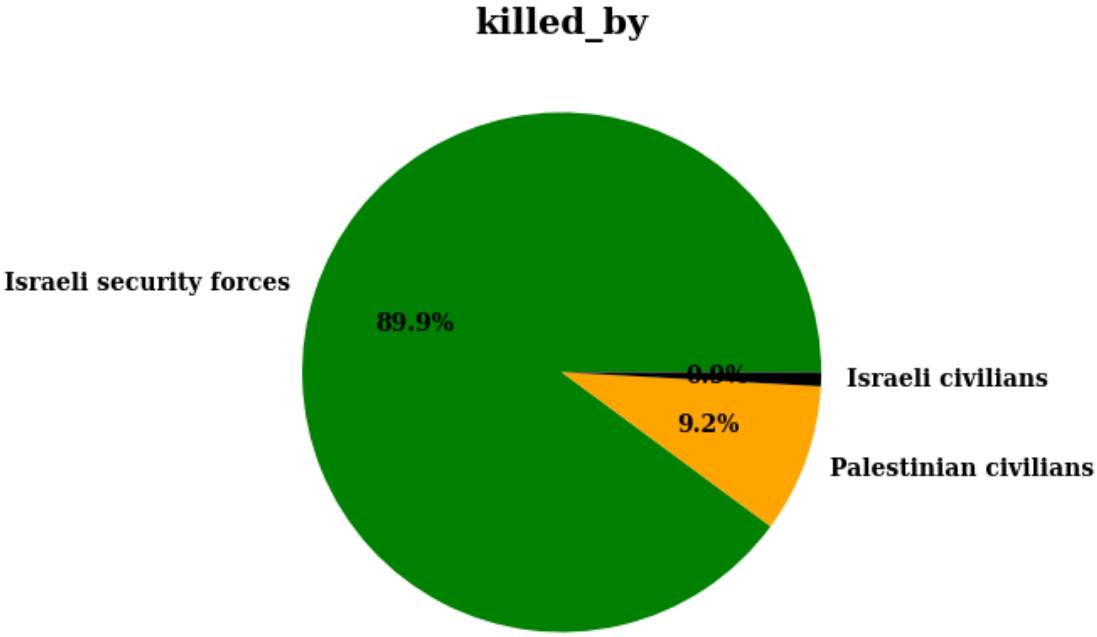
[26]:
```
for i in categorical:
    plt.figure(figsize=(10,5))
    values = df[i].value_counts()
    labels = df[i].value_counts().index
    plt.pie(values, labels=labels, autopct='%1.1f%%', colors=['green',␣
 ↪'orange', 'black', 'blue'], textprops={'weight': 'bold', 'family': 'serif'})
    plt.title(i, size=15, fontdict={'weight': 'bold', 'family': 'serif'})
    plt.show()
```


citizenship
Palestinian
90.7%
0.0%
9.3%
Israeli

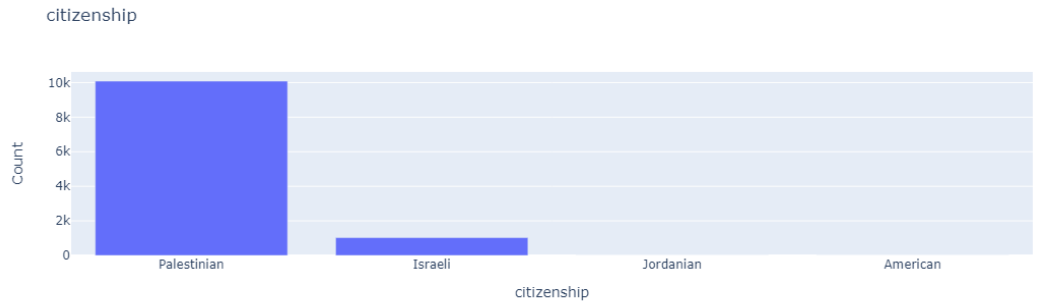

event_location_region
Gaza Strip
69.5%
Israel
24.4%
West Bank

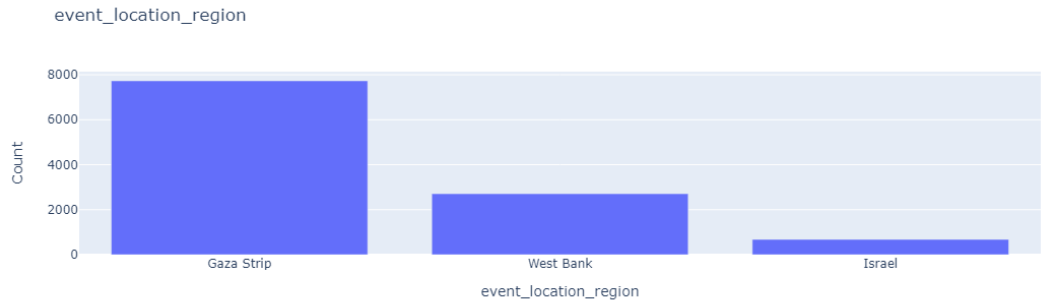

gender
M
87.1%
0.1%
Unknown
12.8%
F

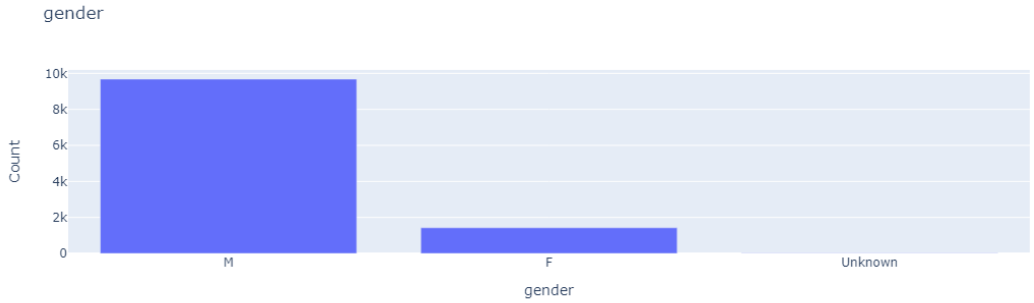

took_part_in_the_hostilities
No
41.9%
1.8%
Object of targeted killing
5.4%
Unknown
31.2%
6.9%
Yes
Israelis
Not Specified

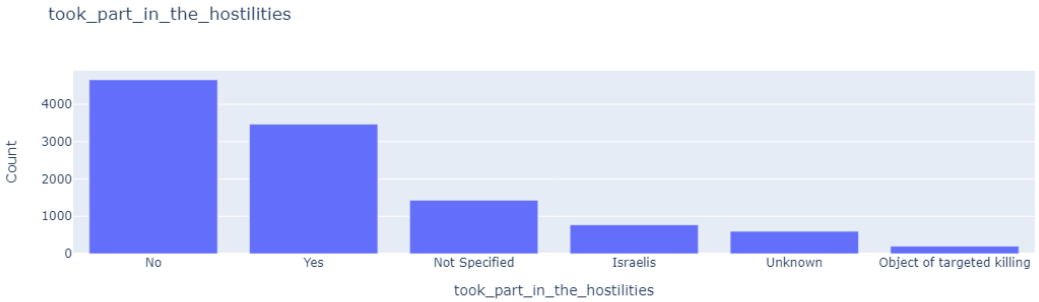

killed_by
Israeli security forces
89.9%
0.9%
Israeli civilians
9.2%
Palestinian civilians

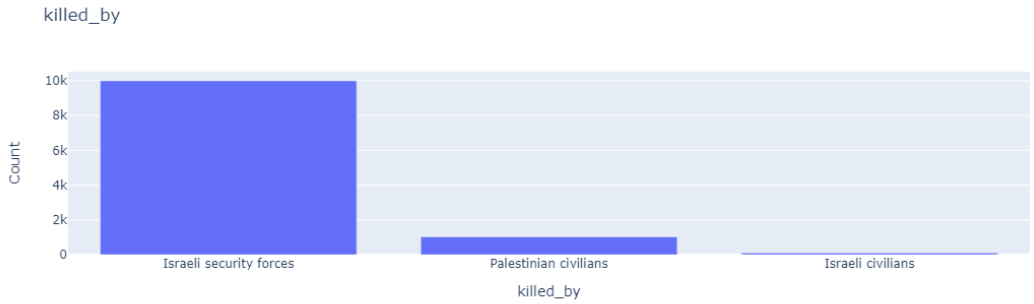
[27]:
```
for i in categorical:
    fig = go.Figure(data=[go.Bar(x=df[i].value_counts().index, y=df[i].
 ↪value_counts())])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Count"
    )
    fig.show()
```











[28]:
```
for i in categorical:
    plt.figure(figsize=(10,5))
    print('Pie plot for:', i)
    fig = px.pie(df,names=i)
    fig.show()
    print('\n')
```

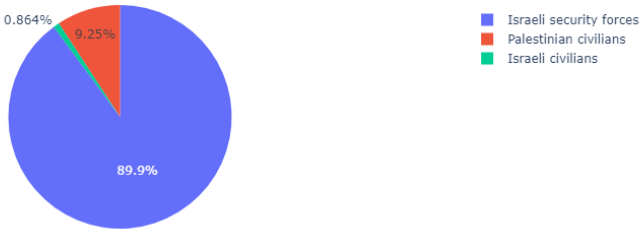
```
Pie plot for: citizenship
```

Pie plot for: event_location_region

Pie plot for: gender

Pie plot for: took_part_in_the_hostilities

Pie plot for: killed_by



<Figure size 1000x500 with 0 Axes>

<Figure size 1000x500 with 0 Axes>

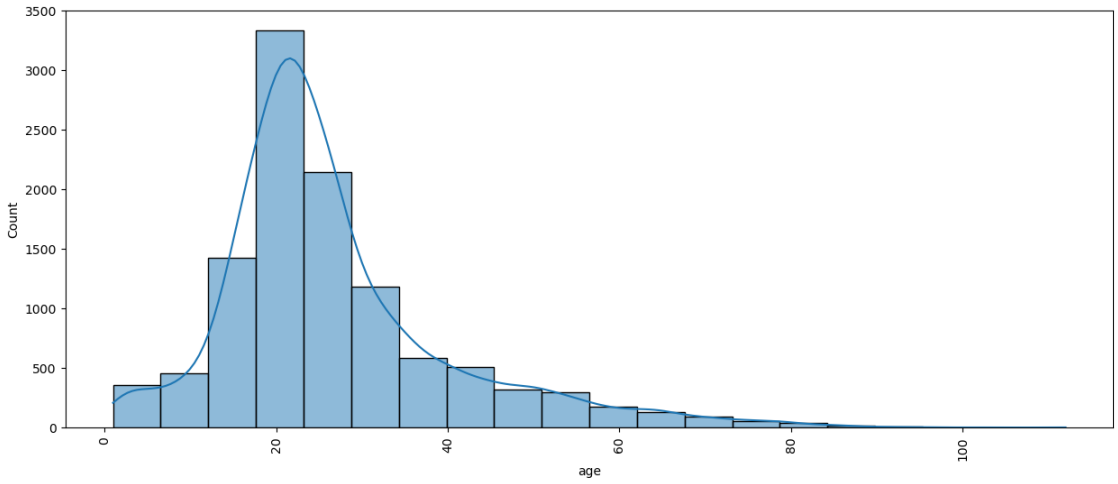
<Figure size 1000x500 with 0 Axes>

<Figure size 1000x500 with 0 Axes>

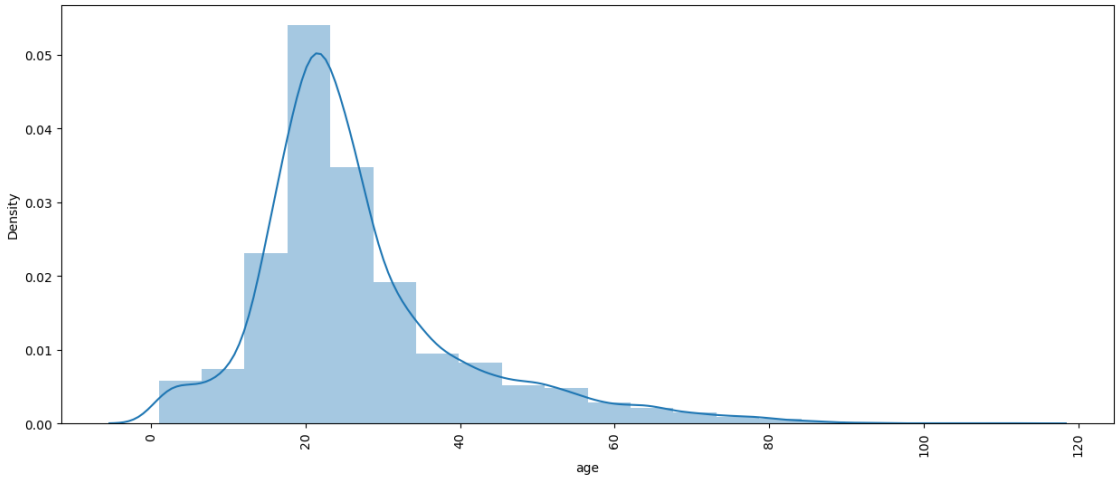
<Figure size 1000x500 with 0 Axes>

[29]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], kde= True, bins =20, palette= 'hls')
```

```
    plt.xticks(rotation = 90)
    plt.show()
```

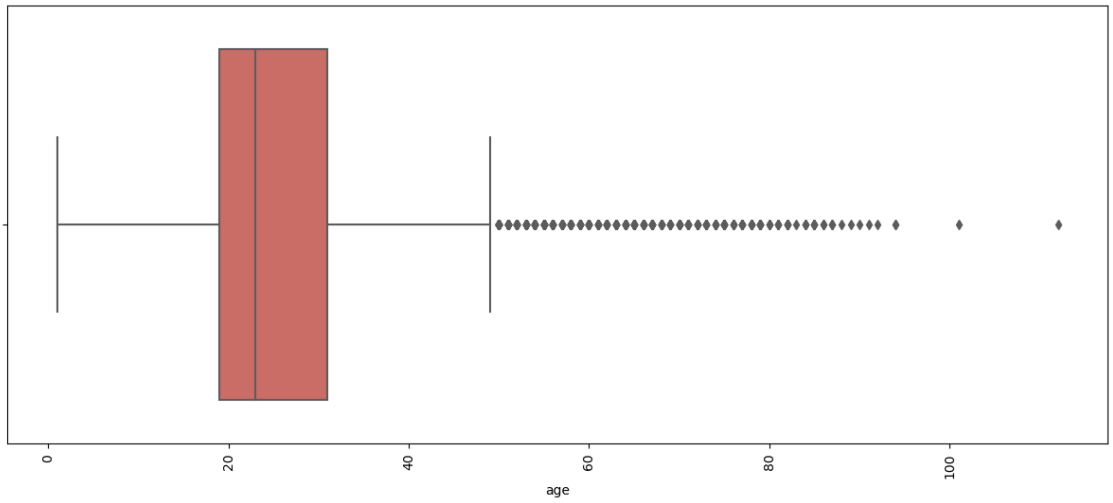


[30]:
```
for i in continuous:
 plt.figure(figsize=(15,6))
 sns.distplot(df[i], kde = True, bins = 20)
 plt.xticks(rotation = 90)
 plt.show()
```

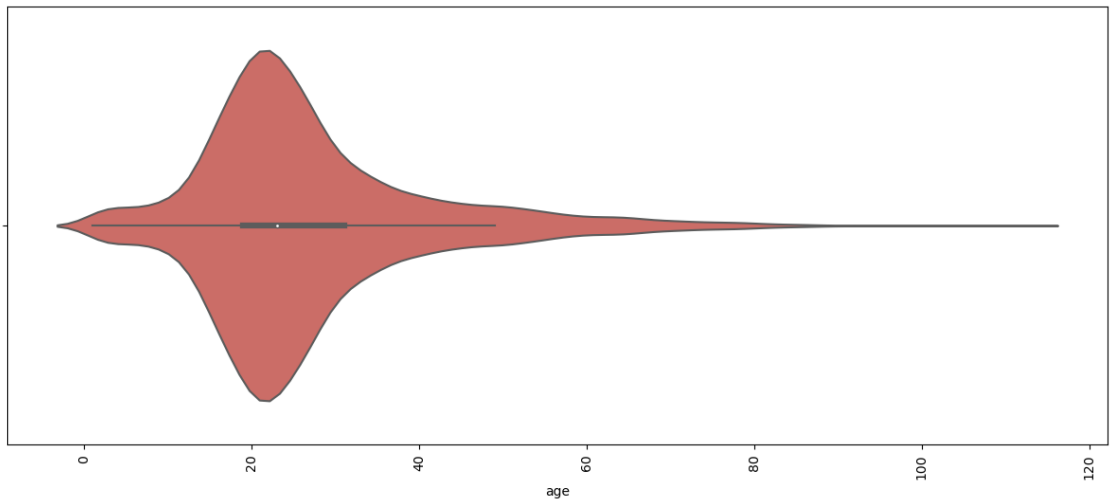


[31]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.boxplot(x=df[i], data = df, palette='hls')
    plt.xticks(rotation = 90)
```

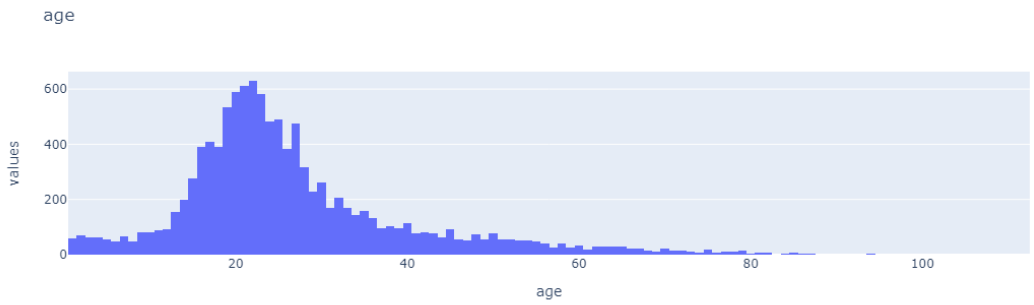
```
    plt.show()
```



[32]:
```
from sklearn.cross_decomposition import PLSCanonical


for i in continuous:
    plt.figure(figsize=(15,6))
    sns.violinplot(x=df[i], data = df, palette= 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

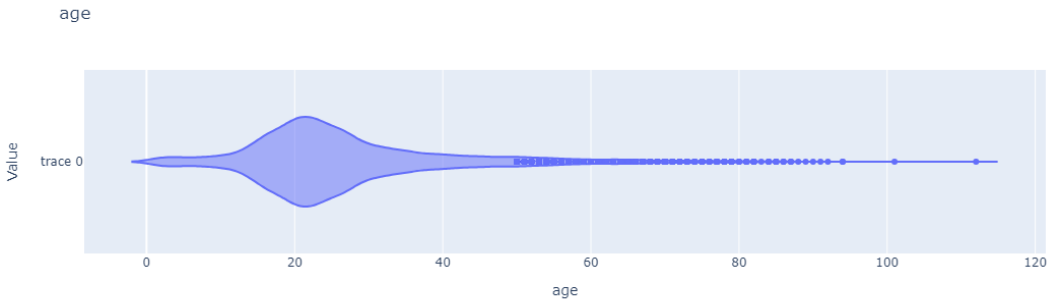


[33]:
```
for i in numrical_columns:
    fig = go.Figure(data=[go.Histogram(x=df[i])])
    plt.figure(figsize=(25,0))

    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="values"
    )
    fig.show()
```




```
<Figure size 2500x0 with 0 Axes>
```

[34]:
```
for i in numrical_columns:
    fig =go.Figure(data=[go.Violin(x=df[i])])
    plt.figure(figsize=(20,5))
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value"
    )
    fig.show()
```



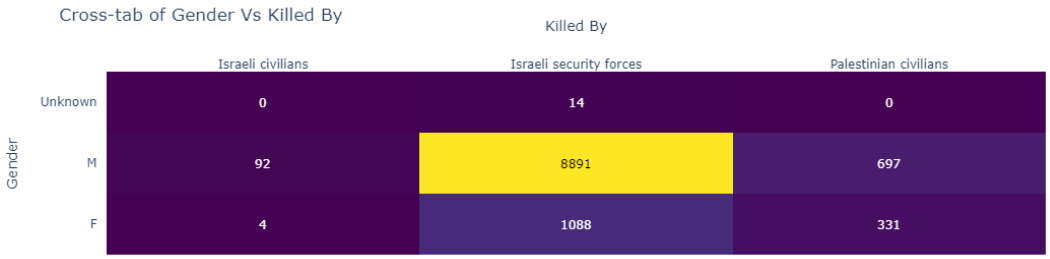
```
<Figure size 2000x500 with 0 Axes>
```

[35]:
```
cross_tab = pd.crosstab(df['gender'],df['killed_by'])
print('Cross-Tabulation of Gender and Killed_by:')
print(cross_tab)
```

```
Cross-Tabulation of Gender and Killed_by:
killed_by  Israeli civilians  Israeli security forces  Palestinian civilians
gender
F                          4                     1088                    331
M                         92                     8891                    697
Unknown                    0                       14                      0
```

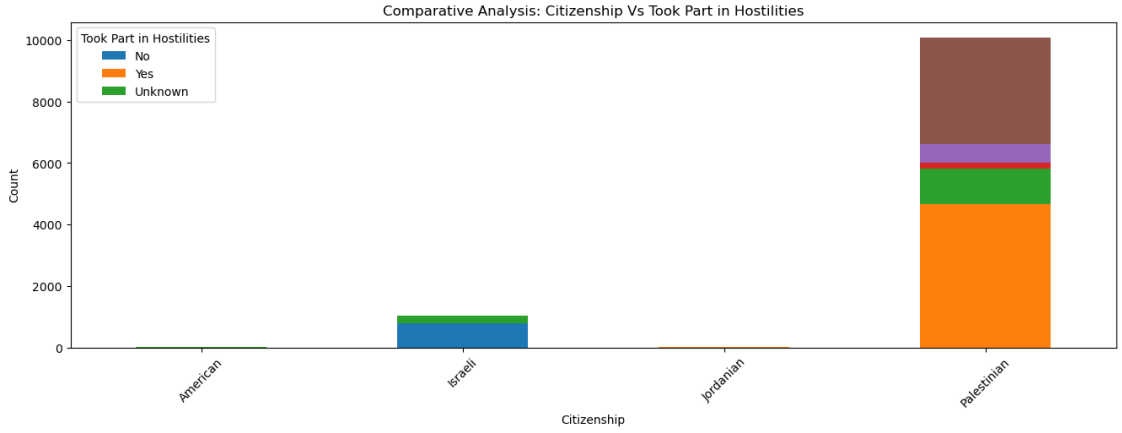
[36]:
```
import plotly.figure_factory as ff
```

[37]:
```
cross_tab = pd.crosstab(df['gender'],df['killed_by'])
plt.figure(figsize=(15,5))


fig = ff.create_annotated_heatmap(
    z=cross_tab.values,
    x=list(cross_tab.columns),
    y=list(cross_tab.index),
    annotation_text=cross_tab.values,
    colorscale='Viridis',
)

fig.update_layout(
    title='Cross-tab of Gender Vs Killed By',
    xaxis_title='Killed By',
    yaxis_title='Gender',
)

fig.show()
```



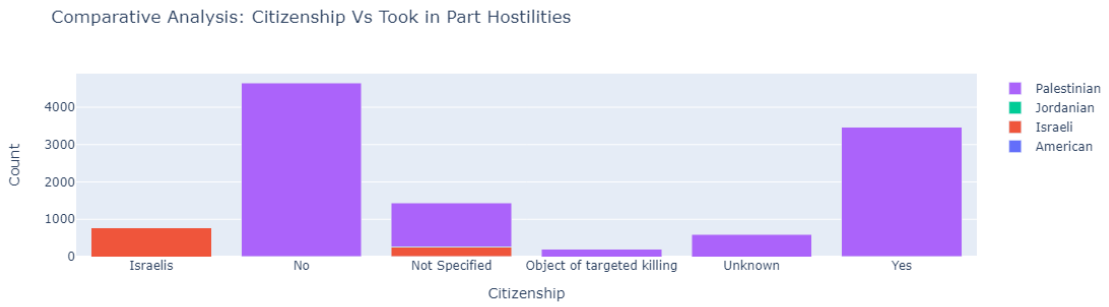
```
<Figure size 1500x500 with 0 Axes>
```

[38]:
```
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).
 ↪size().unstack().fillna(0)
    grouped_data.plot(kind='bar', stacked=True, figsize=(16, 5))

    plt.title('Comparative Analysis: Citizenship Vs Took Part in Hostilities')
    plt.xlabel('Citizenship')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.legend(title='Took Part in Hostilities', labels=['No', 'Yes',␣
 ↪'Unknown'])
    plt.show()

comparative_analysis(df)
```



[39]:
```
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).
 ↪size().unstack().fillna(0)
    plt.figure(figsize=(15,5))



    fig = go.Figure()
    for index, row in grouped_data.iterrows():
        fig.add_trace(go.Bar(
            x=grouped_data.columns,
            y=row.values,
            name=index
```

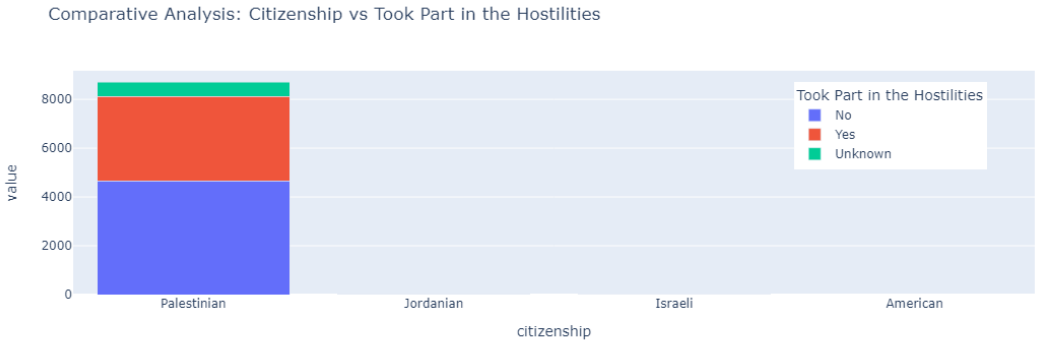
```
        ))


    fig.update_layout(
        title='Comparative Analysis: Citizenship Vs Took in Part Hostilities',
        xaxis=dict(title='Citizenship'),
        yaxis=dict(title='Count'),
        barmode='stack'
    )

    fig.show()

comparative_analysis(df)
```




```
<Figure size 1500x500 with 0 Axes>
```

[40]:
```
import matplotlib.pyplot as plt
import plotly.express as px

def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).
 ↪size().unstack().fillna(0)

    fig = px.bar(
        grouped_data,
        x=grouped_data.index,
        y=['No', 'Yes', 'Unknown'],
        title='Comparative Analysis: Citizenship vs Took Part in the␣
 ↪Hostilities',
        labels={'x': 'Citizenship', 'y': 'Count'},
        height=400
    )

    fig.update_layout(
```

```
        barmode='stack',
        xaxis=dict(categoryorder='total descending'),
        legend_title='Took Part in the Hostilities',
        legend=dict(x=0.75, y=0.95)
    )

    fig.show()


comparative_analysis(df)

plt.figure(figsize=(12, 8))
plt.bar(grouped_data.index, grouped_data['No'], label='No')
plt.bar(grouped_data.index, grouped_data['Yes'], bottom=grouped_data['No'],␣
 ↪label='Yes')
plt.bar(grouped_data.index, grouped_data['Unknown'], bottom=grouped_data['No']␣
 ↪+ grouped_data['Yes'], label='Unknown')

plt.xlabel('Citizenship')
plt.ylabel('Count')
plt.title('Comparative Analysis: Citizenship vs Took Part in Hostilities')
plt.legend(title='Took Part in Hostilities')
plt.xticks(rotation=45)
plt.show()
```



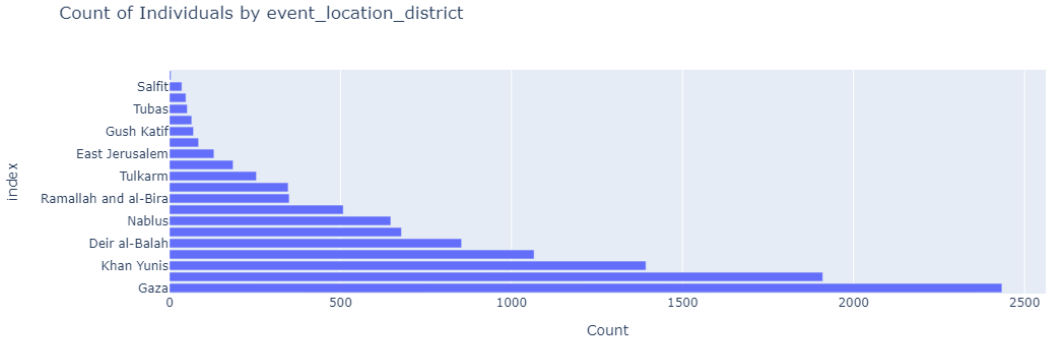

```
---------------------------------------------------------------
NameError                 Traceback (most recent call last)
Cell In[40], line 29
     26 comparative_analysis(df)
     28 plt.figure(figsize=(12, 8))
---> 29 plt.bar(grouped_data.index, grouped_data['No'], label='No')
     30 plt.bar(grouped_data.index, grouped_data['Yes'],␣
 ↪bottom=grouped_data['No'], label='Yes')
```

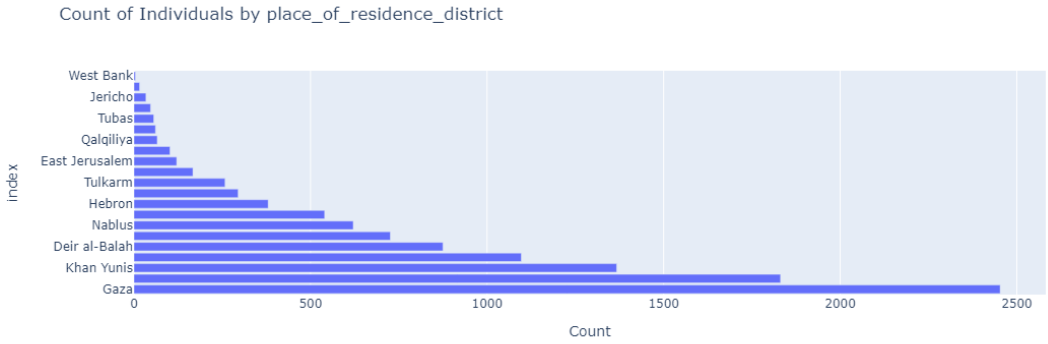
```
     31 plt.bar(grouped_data.index, grouped_data['Unknown'],␣
 ↪bottom=grouped_data['No'] + grouped_data['Yes'], label='Unknown')

NameError: name 'grouped_data' is not defined
```

```
<Figure size 1200x800 with 0 Axes>
```

[ ]:
```
def location_analysis(df):
    location_features = ['event_location_district',␣
 ↪'place_of_residence_district']

    for features in location_features:
        plt.figure(figsize=(15,5))
        sns.countplot(data=df, y=features, order =df[features].value_counts().
 ↪index)
        plt.title(f'Conut of Individuals by {features}')
        plt.xlabel('Count')
        plt.ylabel(features)
        plt.show()

location_analysis(df)
```

[41]:
```
def location_analysis(df):
    location_features = ['event_location_district',␣
 ↪'place_of_residence_district']
    plt.figure(figsize=(15,5))



    for features in location_features:
        counts = df[features].value_counts()
        fig =px.bar(counts, y=counts.index, x=counts.values, orientation ='h',
                    title=f'Count of Individuals by {features}',
                    labels={'x': 'Count', 'y' : features},
                    height=400)

        fig.show()

location_analysis(df)
```






```
<Figure size 1500x500 with 0 Axes>
```

[42]:
```
from wordcloud import WordCloud

def perform_text_analysis(df):
 text = ' '.join(df['notes'].dropna())
 wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(text)
 plt.figure(figsize=(15, 6))
 plt.imshow(wordcloud, interpolation='bilinear')
 plt.axis('off')
 plt.title('Word Cloud - Notes')
 plt.show()
perform_text_analysis(df)
```

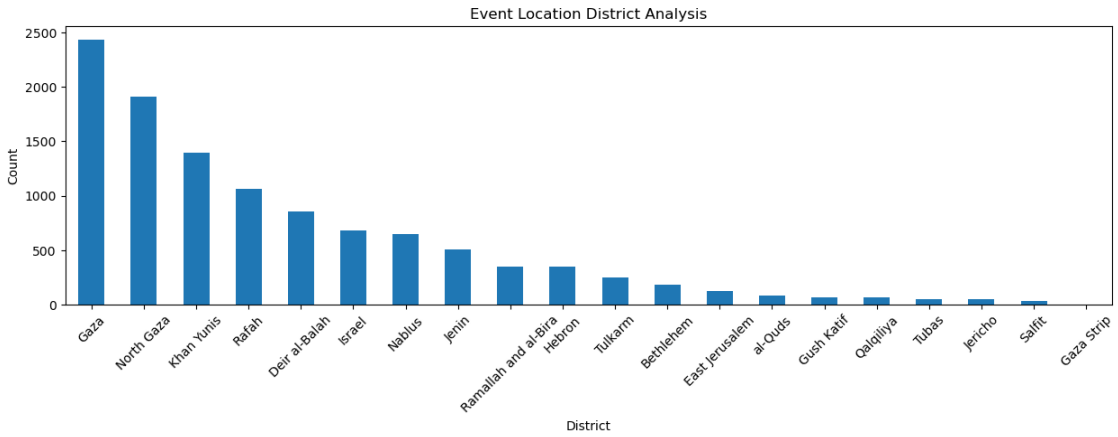
```
---------------------------------------------------------------
ModuleNotFoundError       Traceback (most recent call last)
Cell In[42], line 1
----> 1 from wordcloud import WordCloud
```

```
      3 def perform_text_analysis(df):
      4   text = ' '.join(df['notes'].dropna())

ModuleNotFoundError: No module named 'wordcloud'
```

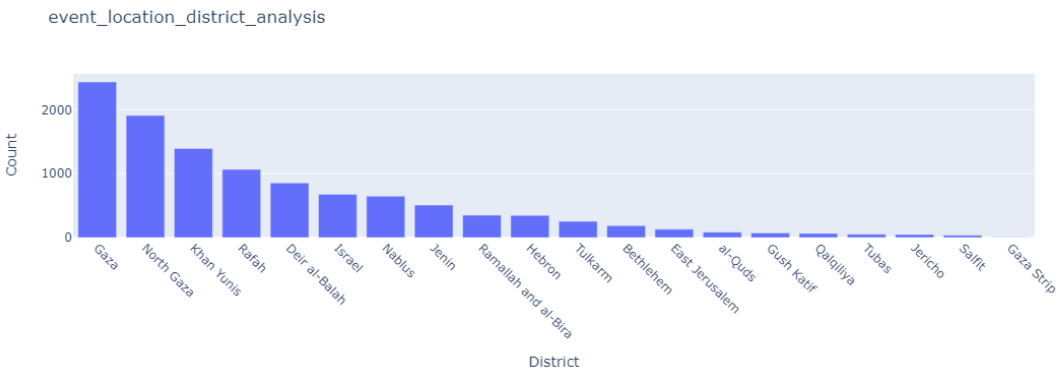
[43]:
```
def event_location_analysis(df):
    plt.figure(figsize=(15,5))

    fig = px.scatter_geo(df,
                         locations='event_location',
                         locationmode='country names',
                         title='Event Location Distribution',
                         projection='natural earth')
    fig.show()

event_location_analysis(df)
```


```
<Figure size 1500x500 with 0 Axes>
```

[44]:
```
def event_location_analysis(df):
    plt.figure(figsize=(15,4))
    df['event_location_district'].value_counts().plot(kind='bar')
    plt.title('Event Location District Analysis')
    plt.xlabel('District')
    plt.ylabel('Count')
    plt.xticks(rotation = 45)
    plt.show()


event_location_analysis(df)
```

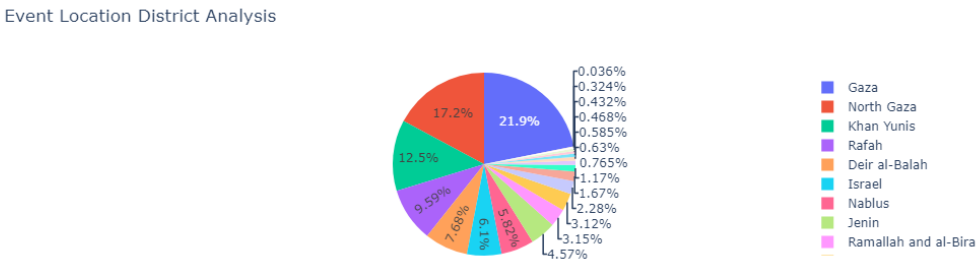


[45]:
```python
def event_location_district_analysis(df):
    plt.figure(figsize=(12,8))
    district_counts =df['event_location_district'].value_counts()

    fig = px.bar(district_counts,
                 x=district_counts.index,
                 y=district_counts.values,
                 title='event_location_district_analysis',
                 labels={'x': 'District', 'y':'Count'},
                 height=400)

    fig.update_layout(xaxis_title='District', yaxis_title='Count')
    fig.update_xaxes(tickangle=45)

    fig.show()

event_location_district_analysis(df)
```



```
<Figure size 1200x800 with 0 Axes>
```

[46]:
```
def event_location_district_analysis(df):
    plt.figure(figsize=(12,8))

    district_counts = df ['event_location_district'].value_counts()

    fig = px.pie(district_counts,
                 names=district_counts.index,
                 values=district_counts.values,
                 title='Event Location District Analysis')

    fig.show()

event_location_district_analysis(df)
```




```
<Figure size 1200x800 with 0 Axes>
```

[47]:
```
def name_analysis(df):

    missing_name_count = df['name'].isnull().sum()

    invalid_name = df [~df['name'].str.match(r'^[A-Za-z\s\'.-]+$')]['name']

    return missing_name_count, invalid_name

missing_name_count, invalid_name = name_analysis(df)

print("Number of Missing Names :", missing_name_count)
```

```
Number of Missing Names : 0
```

[48]:
```
print("Invalid Name")
invalid_name
```

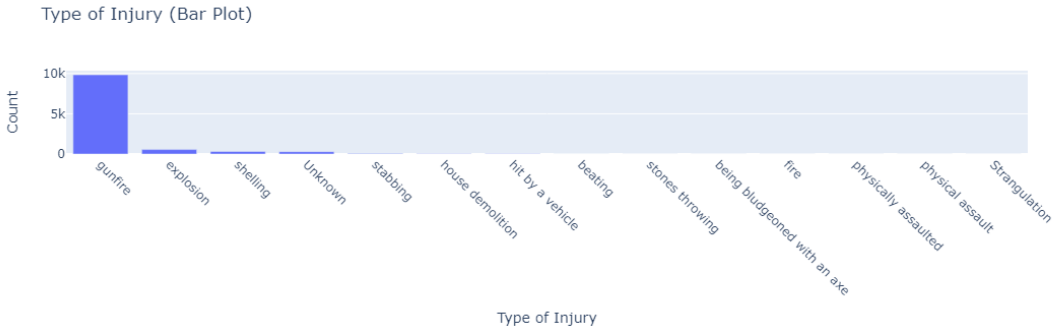
```
Invalid Name
```

[48]:
```
62          Suhaib 'Adnan Jum'ah Musa (al-Ghul)
145                              Lucy (Leah) Dee
196                                   Asher
197                                  Alter
251          Sidqi Sadiq Fayeq Jabur (Zakarnah)
                            …
9823                            Hagai (Haim) Lev
10075    Jamal Tawfiq 'Issa Turkman ('Ar'arawi)
10154                                   Jamil
10852               Mahmoud a-Shuli (Abu Hanud)
10953                  Rechavam Ze'evy (Gandhi)
Name: name, Length: 115, dtype: object
```

[49]:
```
def place_of_resisdence_analysis(df):
    plt.figure(figsize=(12,8))

    residence_counts = df['place_of_residence'].value_counts()

    fig = px.bar(residence_counts,
                 x=residence_counts.index,
                 y=residence_counts.values,
                 title='Place of Residence Analysis',
                 labels={'x': 'Place of Residence', 'y': 'Count'},
    height=400)

    fig.update_layout(xaxis_title = 'Place of Residence', yaxis_title='Count')
    fig.update_xaxes(tickangle=45)

    fig.show()

place_of_resisdence_analysis(df)
```

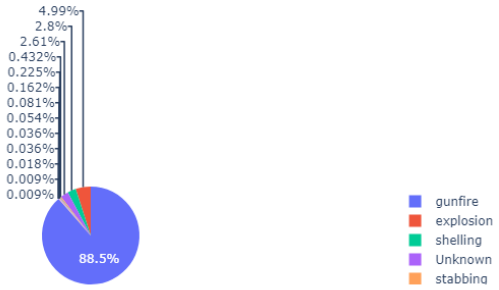
<Figure size 1200x800 with 0 Axes>

[50]:
```
def type_of_injury_analysis_bar(df):
    plt.figure(figsize=(12,8))
    injury_count = df['type_of_injury'].value_counts()

    fig = px.bar(injury_count,
                 x=injury_count.index,
                 y=injury_count.values,
                 title='Type of Injury (Bar Plot)',
                 labels={'x': 'Type of Injury', 'y': 'Count'},
                 )

    fig.update_layout(xaxis_title = 'Type of Injury', yaxis_title ='Count')
    fig.update_xaxes(tickangle = 45)

    fig.show()

type_of_injury_analysis_bar(df)
```



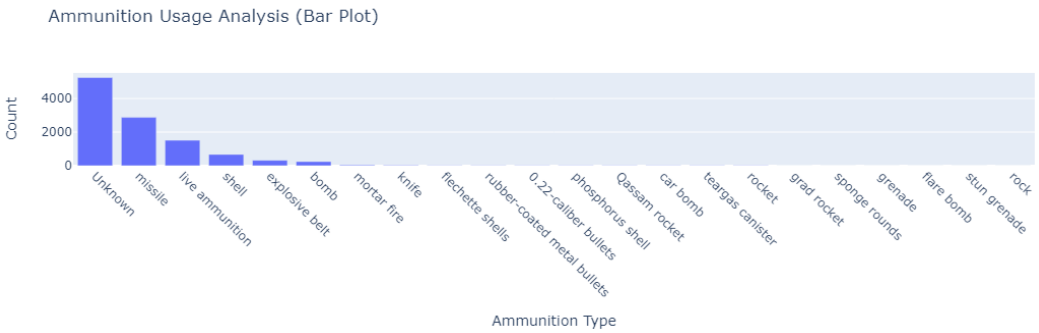

```
<Figure size 1200x800 with 0 Axes>
```

[51]:
```
def type_of_injury_analysis_bar(df):
    plt.figure(figsize=(18,8))
    injury_count =df ['type_of_injury'].value_counts()

    fig = px.pie(injury_count,
                 names=injury_count.index,
                 values=injury_count.values)

    fig.show()

type_of_injury_analysis_bar(df)
```

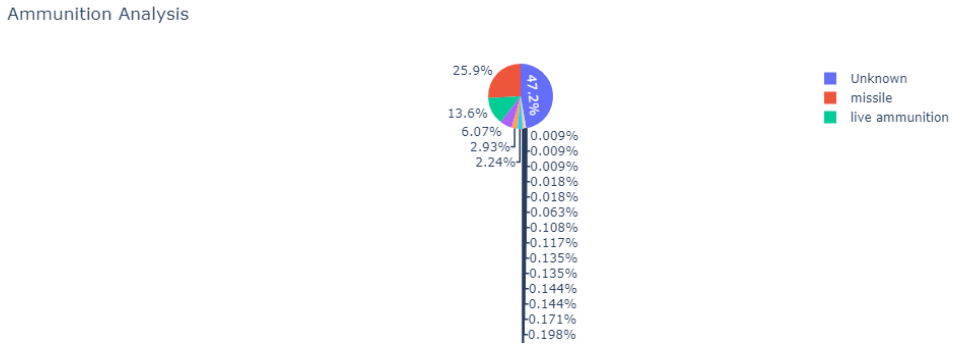


<Figure size 1800x800 with 0 Axes>

[52]:
```
def ammunition_analysis_bar(df):

    ammo_counts = df ['ammunition'].value_counts()

    fig = px.bar(ammo_counts,
                 x=ammo_counts.index,
                 y=ammo_counts.values,
                 title='Ammunition Usage Analysis (Bar Plot)',
                 labels={'x': 'Ammunitation Type', 'y': 'Count'},
                 )

    fig.update_layout(xaxis_title = 'Ammunition Type', yaxis_title ='Count')
    fig.update_xaxes(tickangle=45)

    fig.show()

ammunition_analysis_bar(df)
```



[53]:
```
def ammunition_analysis (df):
    plt.figure(figsize=(7,7))

    ammo_counts = df['ammunition'].value_counts()

    fig = px.pie(ammo_counts,
                 names=ammo_counts.index,
                 values=ammo_counts.values,
                 title='Ammunition Analysis',
                 )


    fig.show()

ammunition_analysis(df)
```



<Figure size 700x700 with 0 Axes>

[54]:
```
import nltk
from nltk import FreqDist
from nltk.tokenize import word_tokenize
```

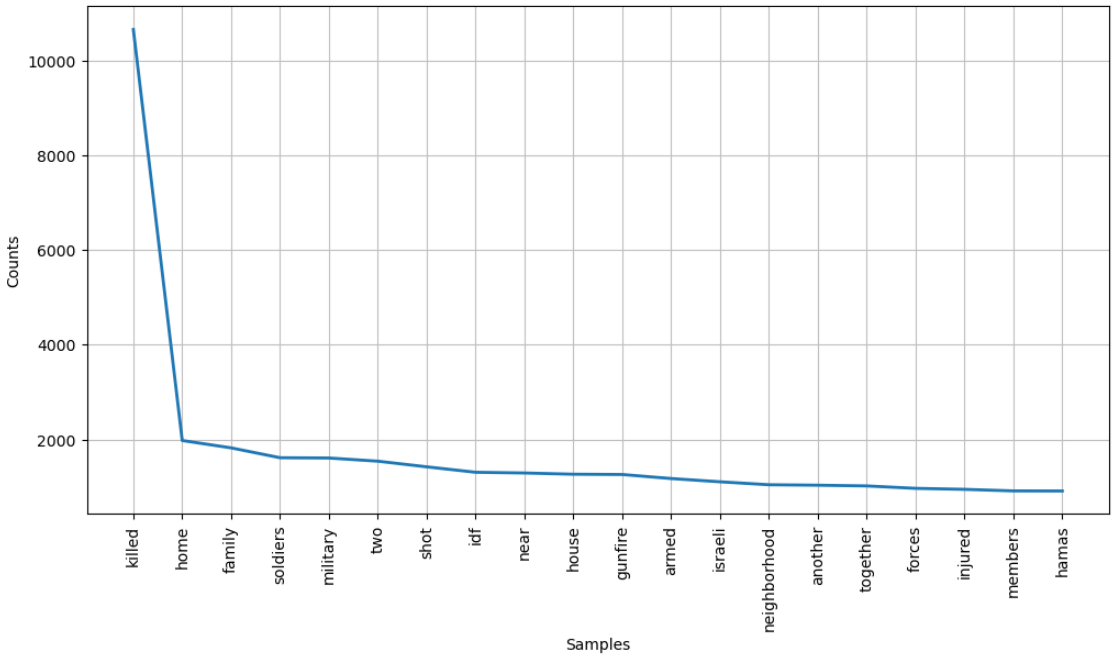
[55]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk import FreqDist
import matplotlib.pyplot as plt

def notes_word_frequency_analysis(df):
    plt.figure(figsize=(15,5))

    nltk.data.path.append("E:\\NLTK PY")
    nltk.download("vader_lexicon")
```

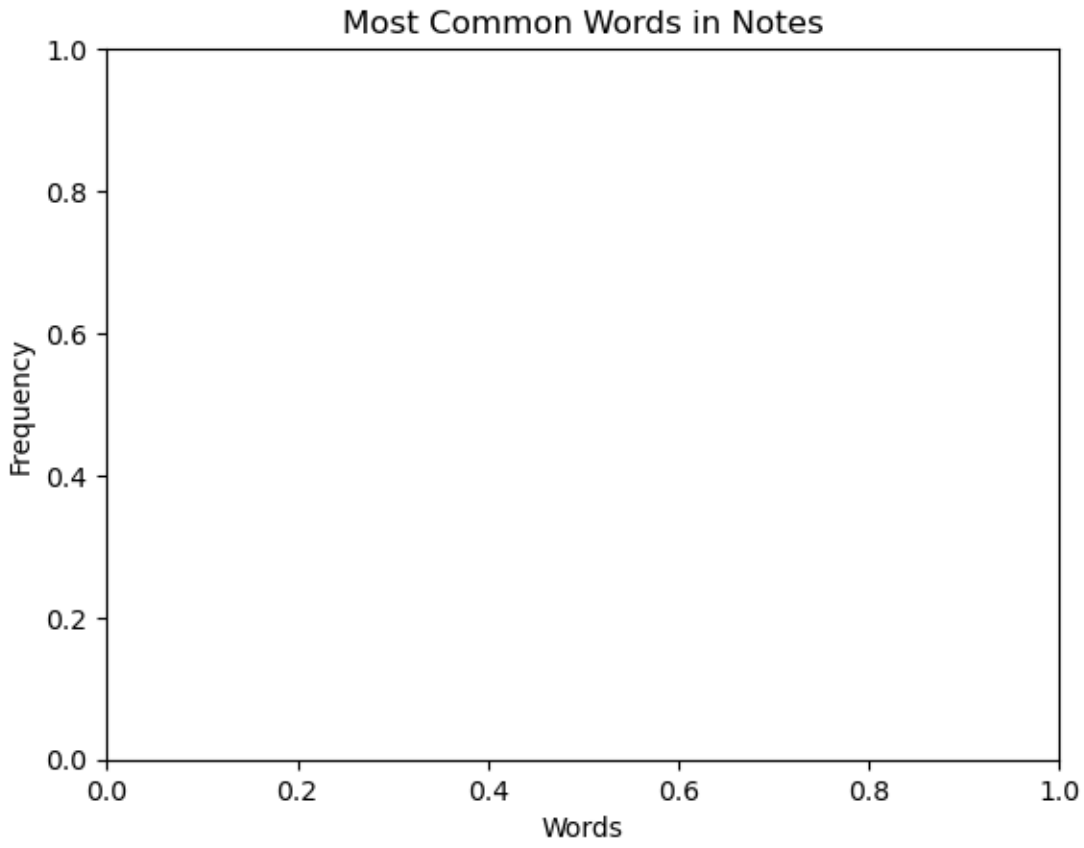
```
    all_notes = ' '.join(df['notes'].dropna())
    words = word_tokenize(all_notes)
    stop_words = set(stopwords.words('english'))
    filtered_words = [word.lower() for word in words if word.isalpha() and word.
 ↪lower() not in stop_words]
    word_freq = FreqDist(filtered_words)
    plt.figure(figsize=(12, 6))
    word_freq.plot(20, cumulative=False)
    plt.title('Most Common Words in Notes')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.show()
```

[56]:
```
notes_word_frequency_analysis(df)
plt.figure(figsize=(15,5))
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\mrala\AppData\Roaming\nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
<Figure size 1500x500 with 0 Axes>
```





[56]:
```
<Figure size 1500x500 with 0 Axes>
```

```
<Figure size 1500x500 with 0 Axes>
```

[57]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist


def notes_word_frequency_analysis(df):
    all_notes = ' '.join(df['notes'].dropna())
    words = word_tokenize(all_notes)
    stopwords = nltk.corpus.stopwords.words('english')
    filtered_words = [word.lower() for word in words if word.isalpha() and word.
 ↪lower() not in stopwords]
    word_freq = FreqDist(filtered_words)

    print("Most Common Words:")
    print(word_freq.most_common(20))
```

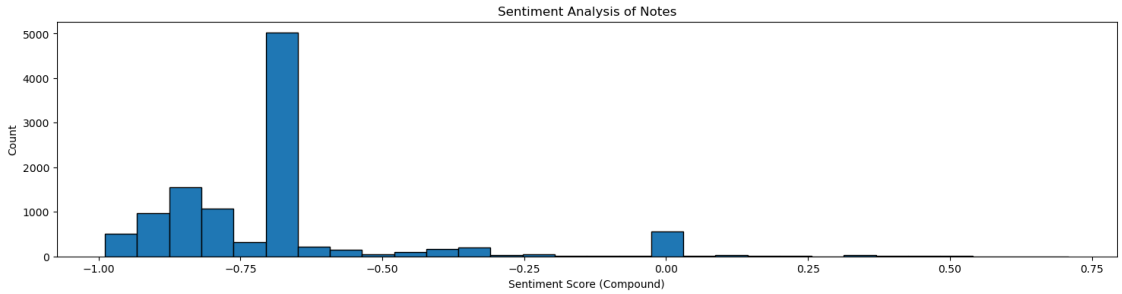
```
notes_word_frequency_analysis(df)
```

```
Most Common Words:
[('killed', 10653), ('home', 1981), ('family', 1826), ('soldiers', 1618),
('military', 1612), ('two', 1545), ('shot', 1426), ('idf', 1311), ('near',
1296), ('house', 1270), ('gunfire', 1264), ('armed', 1179), ('israeli', 1110),
('neighborhood', 1048), ('another', 1038), ('together', 1023), ('forces', 972),
('injured', 951), ('members', 918), ('hamas', 916)]
```

[58]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

[59]:
```
def sentiment_analysis(df):
    sid = SentimentIntensityAnalyzer()
    df['sentiment_score'] = df['notes'].apply(lambda x: sid.
 ↪polarity_scores(str(x))['compound'])

    plt.figure(figsize=(18, 4))
    plt.hist(df['sentiment_score'], bins=30, edgecolor='black')
    plt.title('Sentiment Analysis of Notes')
    plt.xlabel('Sentiment Score (Compound)')
    plt.ylabel('Count')
    plt.show()
```

[60]:
```
sentiment_analysis(df)
```



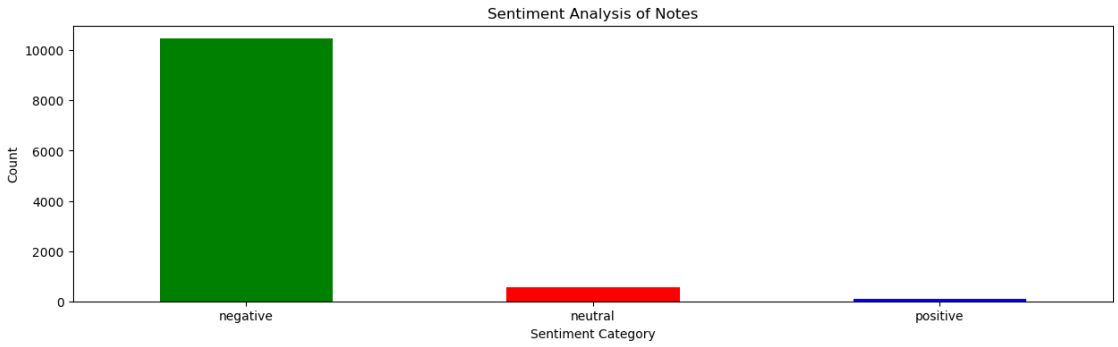
[61]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

[62]:
```
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer

def sentiment_analysis(df):
    sid = SentimentIntensityAnalyzer()
    df['sentiment_score'] = df['notes'].apply(lambda x: sid.
 ↪polarity_scores(str(x))['compound'])

    plt.figure(figsize=(18, 4))
```

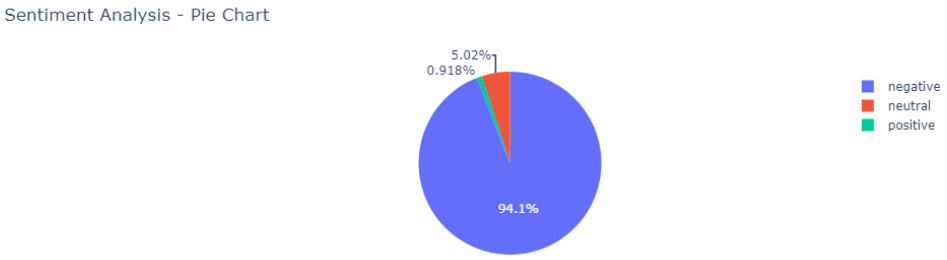
```
    plt.hist(df['sentiment_score'], bins=30, edgecolor='black')
    plt.title('Sentiment Analysis of Notes')
    plt.xlabel('Sentiment Score (Compound)')
    plt.ylabel('Count')
    plt.show()
```

[63]:
```
import pandas as pd
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk

# Ensure that NLTK data is downloaded
nltk.download("vader_lexicon")

sid = SentimentIntensityAnalyzer()

df['sentiment_score'] = df['notes'].apply(lambda x: sid.
 ↪polarity_scores(str(x))['compound'])
df['sentiment_category'] = df['sentiment_score'].apply(lambda score: 'positive'␣
 ↪if score > 0 else ('negative' if score < 0 else 'neutral'))

plt.figure(figsize=(15, 4))
df['sentiment_category'].value_counts().plot(kind='bar', color=['green', 'red',␣
 ↪'blue'])
plt.title('Sentiment Analysis of Notes')
plt.xlabel('Sentiment Category')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\mrala\AppData\Roaming\nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```



[64]:
```
# Assuming df is your DataFrame
sid = SentimentIntensityAnalyzer()

def sentiment_analysis_pie_plotly(df):
    df['sentiment_score'] = df['notes'].apply(lambda x: sid.
 ↪polarity_scores(str(x))['compound'])
    df['sentiment_category'] = df['sentiment_score'].apply(lambda score:␣
 ↪'positive' if score > 0 else ('negative' if score < 0 else 'neutral'))
    sentiment_distribution = df['sentiment_category'].value_counts()

    fig = px.pie(
        values=sentiment_distribution.values,
        names=sentiment_distribution.index,
        title='Sentiment Analysis - Pie Chart'
    )
    fig.show()

# Call the function
sentiment_analysis_pie_plotly(df)
```



[ ]:
```
def generate_wordcloud(sentiment_category, notes):
    text = ' '.join(notes)
    wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for {sentiment_category.capitalize()} Sentiment')
    plt.show()

def generate_wordclouds_for_sentiments(df):
    for sentiment_category in ['positive', 'negative', 'neutral']:
```

```
        notes_for_sentiment = df[df['sentiment_category'] ==␣
 ↪sentiment_category]['notes']
generate_wordcloud(sentiment_category, notes_for_sentiment)
```

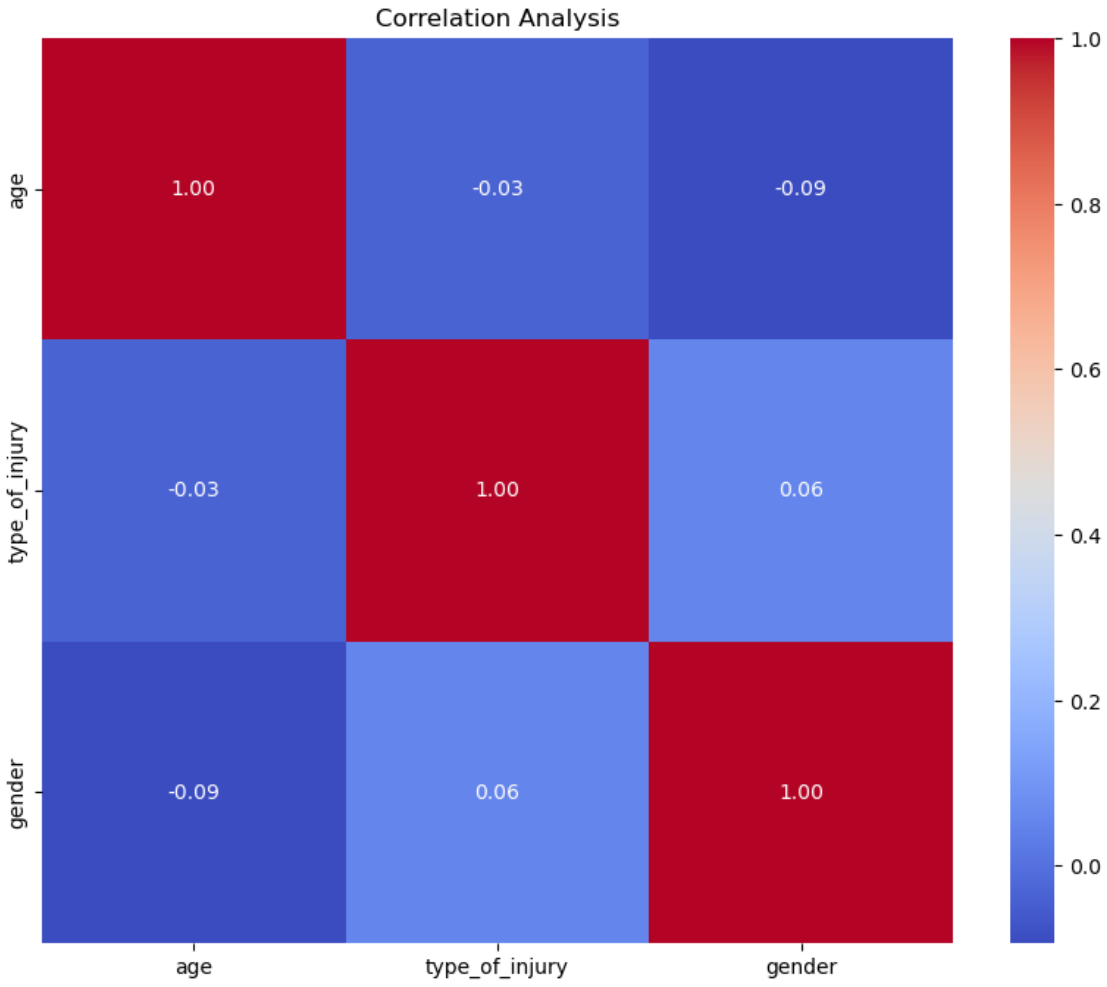
[ ]:
```
df['date_of_event'] = pd.to_datetime(df['date_of_event'])
```

[ ]:
```
def time_based_analysis(df):

    events_by_date = df.groupby(df['date_of_event'].dt.date).size()


    plt.figure(figsize=(15,4))
    events_by_date.plot()
    plt.title('Events Over Time')
    plt.xlabel('Date')
    plt.ylabel('Number of Events')
    plt.show()

time_based_analysis(df)
```

[ ]:
```
def time_based_analysis_plotly(df):

    events_by_date = df.groupby(df['date_of_event'].dt.date).size().
 ↪reset_index()
    events_by_date.columns = ['Date', 'Event Count']

    fig = px.line(events_by_date, x='Date', y='Event Count', title='Event Over␣
 ↪Time')
    fig.show()

time_based_analysis_plotly(df)
```

[66]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def correlation_analysis(df):
    selected_columns = ['age', 'type_of_injury', 'gender',␣
 ↪'took_part_in_the_hostilities']
    selected_df = df[selected_columns]

    # Replace 'Not Specified' with NaN in relevant columns
    selected_df['age'] = pd.to_numeric(selected_df['age'], errors='coerce')
    selected_df['type_of_injury'] = selected_df['type_of_injury'].
 ↪astype('category').cat.codes
    selected_df['gender'] = selected_df['gender'].astype('category').cat.codes
```

```
    selected_df['took_part_in_the_hostilities'] =␣
 ↪selected_df['took_part_in_the_hostilities']

    correlation_matrix = selected_df.corr()

    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title('Correlation Analysis')
    plt.show()

# Call the function for correlation analysis
correlation_analysis(df)
```



[67]:
```
from sklearn.cluster import KMeans
```

[68]:
```
def clustering_and_segmentation(df):
    plt.figure(figsize=(15,5))

    features = ['age', 'gender_numeric', 'type_of_injury_numeric']
    selected_df = df[['age', 'gender', 'type_of_injury']].copy()

    selected_df['gender_numeric'] = selected_df['gender'].astype('category').
 ↪cat.codes
    selected_df['type_of_injury_numeric'] = selected_df['type_of_injury'].
 ↪astype('category').cat.codes

    selected_df.dropna(subset=features, inplace=True)

    kmeans = KMeans(n_clusters=3, random_state=42)
    selected_df['cluster'] = kmeans.fit_predict(selected_df[features])

    plt.figure(figsize=(15, 4))
    sns.scatterplot(x='age', y='type_of_injury_numeric', hue='cluster',␣
 ↪data=selected_df)

    plt.title('Clustering And Segmentation')
    plt.xlabel('Age')
    plt.ylabel('Type of Injury (Numeric)')
    plt.show()

clustering_and_segmentation(df)
```

```
<Figure size 1500x500 with 0 Axes>
```

1.0.1 ————————————————————————————- Thank You !!! ————————————————————————————

[69]:
```
# Projet by: Uvesh Ahmad
# Connect Me : https://uvesh-ahmad.github.io/uvesh.ah
# Linkedin : https://www.linkedin.com/in/uvesh-ahmad-a2aa6816a
# Join Github Repository : https://github.com/Uvesh-Ahmad
```