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

In [1]:
```
# Project by: Prof. Nirmal Gaud
# Contact: ds.ml.projects.sessions.1@gmail.com
# WhatssApp Group (Join for ML/DL Projects): https://chat.whatsapp.com/BQ0vLtxjVS3I1M9Yd
```

In [2]:
```
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
```

In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df = pd.read_csv('fatalities_israel_palestine.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	name	date_of_event	age	citizenship	event_location	event_location_district	event_location_region	d
0	'Abd a-Rahman Suleiman Muhammad Abu Daghash	24-09-2023	32.0	Palestinian	Nur Shams R.C.	Tulkarm	West Bank	
1	Usayed Farhan Muhammad 'Ali Abu 'Ali	24-09-2023	21.0	Palestinian	Nur Shams R.C.	Tulkarm	West Bank	
2	'Abdallah 'Imad Sa'ed Abu Hassan	22-09-2023	16.0	Palestinian	Kfar Dan	Jenin	West Bank	
3	Durgham Muhammad	20-09-2023	19.0	Palestinian	'Aqbat Jaber R.C.	Jericho	West Bank	

	Yihya al-Akhras							
4	Raafat 'Omar Ahmad Khamaisah	19-09-2023	15.0	Palestinian	Jenin R.C.		Jenin	West Bank

In [6]:
```
df.tail()
```

Out[6]:

	name	date_of_event	age	citizenship	event_location	event_location_district	event_location_region
11119	Binyamin Herling	19-10-2000	64.0	Israeli	Nablus	Nablus	West Bank
11120	Farid Musa 'Issa a-Nesasreh	17-10-2000	28.0	Palestinian	Beit Furik	Nablus	West Bank
11121	Hillel Lieberman	07-10-2000	36.0	Israeli	Nablus	Nablus	West Bank
11122	Fahed Mustafa 'Odeh Baker	07-10-2000	21.0	Palestinian	Bidya	Salfit	West Bank
11123	Wichlav Zalsevsky	02-10-2000	24.0	Israeli	Masha	Salfit	West Bank

In [7]:
```
df.shape
```

Out[7]:
```
(11124, 16)
```

In [8]:
```
df.columns
```

Out[8]:
```
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')
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
7
```

In [10]:
```
df = df.drop_duplicates()
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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
```

In [12]:
```
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
```

In [13]:
```
df.describe()
```

Out[13]:

	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

In [14]:
```
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)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
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
```

In [16]:
```
df['date_of_event'] = pd.to_datetime(df['date_of_event'])
df['date_of_death'] = pd.to_datetime(df['date_of_death'])
```

In [17]:
```
df.nunique()
```

Out[17]:
```
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
```

In [18]:
```
object_columns = df.select_dtypes(include=['object']).columns
print("Object type columns:")
print(object_columns)

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

In [19]:
```
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 df[column].dtype in ['int64', 'float64']:
            if df[column].nunique() < 10:
                discrete_features.append(column)
            else:
                continuous_features.append(column)

    return categorical_features, non_categorical_features, discrete_features, continuous
```

In [20]:
```
categorical, non_categorical, discrete, continuous = classify_features(df)
```

In [21]:
```
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_t
he_hostilities', 'killed_by']
Non-Categorical Features: ['name', 'event_location', 'event_location_district', 'place_o
f_residence', 'place_of_residence_district', 'type_of_injury', 'ammunition', 'notes']
Discrete Features: []
Continuous Features: ['age']
```

In [22]:
```
for i in categorical:
    print(i,':')
    print(df[i].unique())
    print('\n')
```

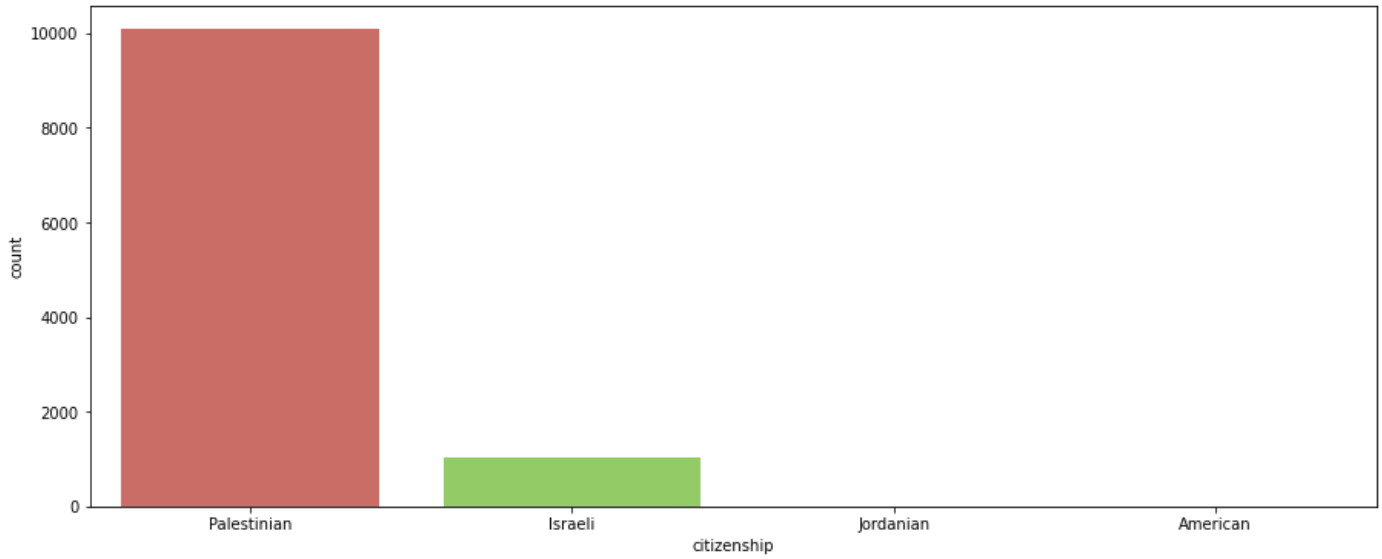
```
citizenship :
['Palestinian' 'Israeli' 'Jordanian' 'American']


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']
```

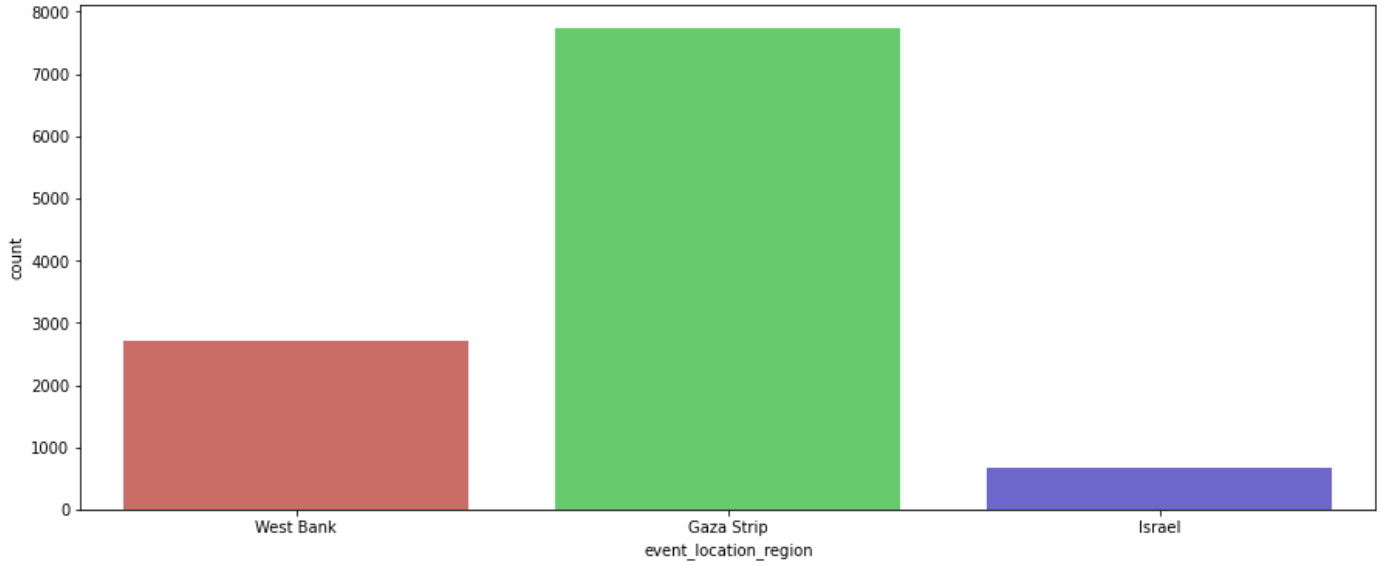
In [23]:
```
for i in categorical:
    print(i,':')
    print(df[i].value_counts())
    print('\n')
```

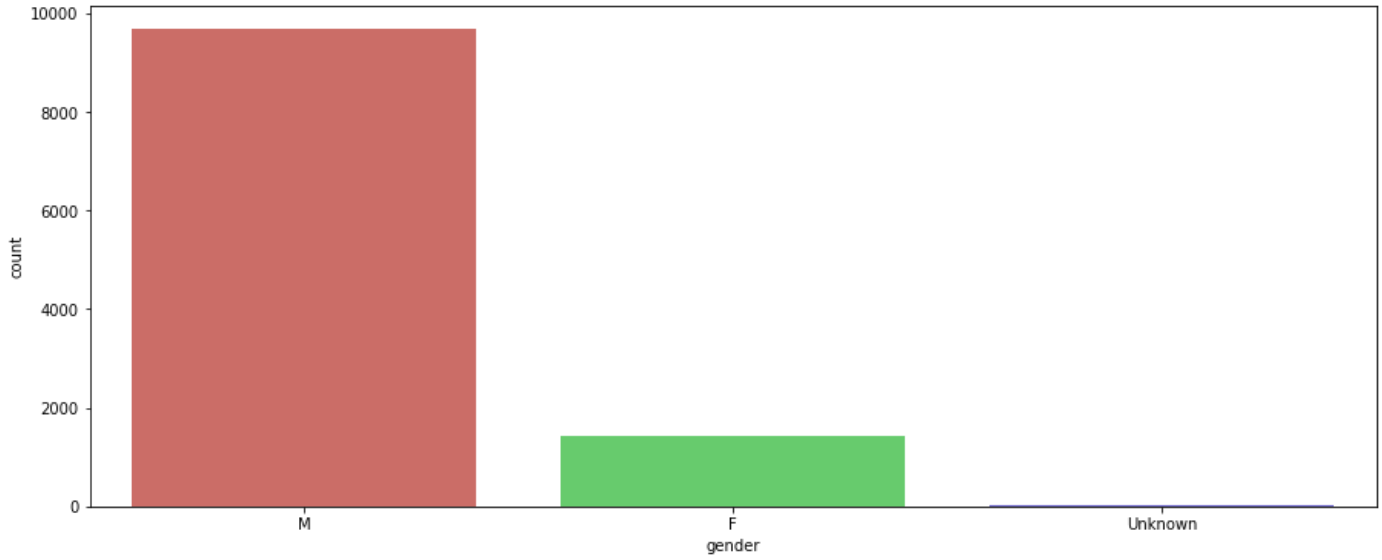
```
citizenship :
Palestinian    10085
Israeli         1029
Jordanian          2
American           1
```

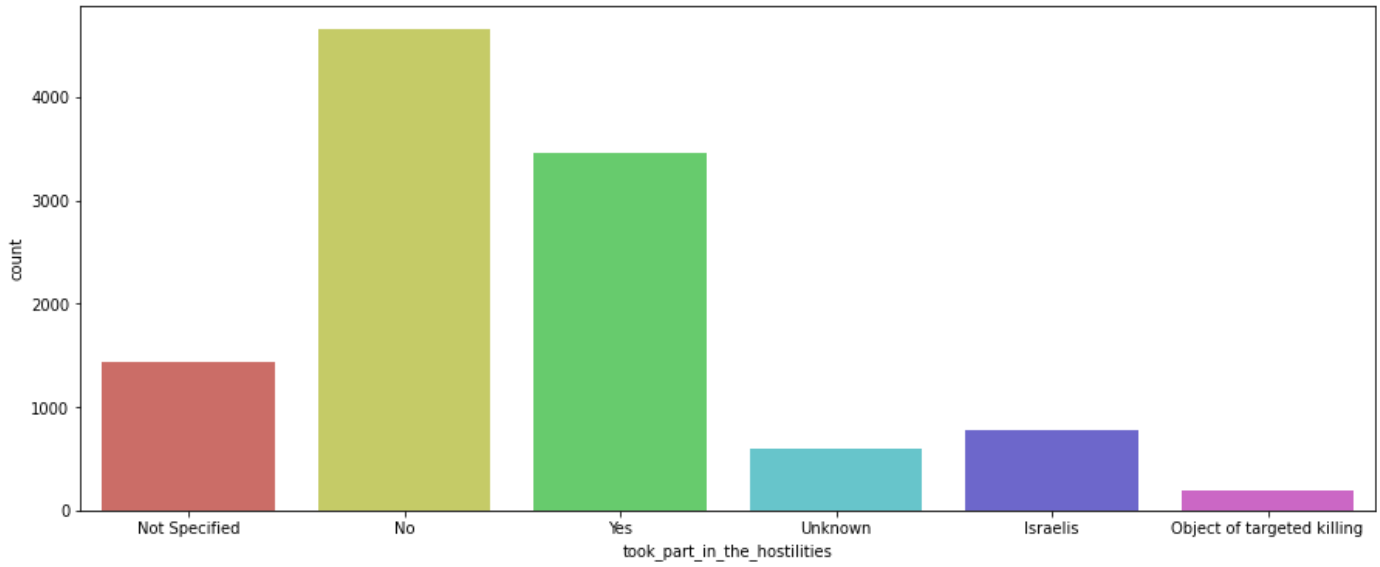
```
Name: citizenship, dtype: int64


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
```

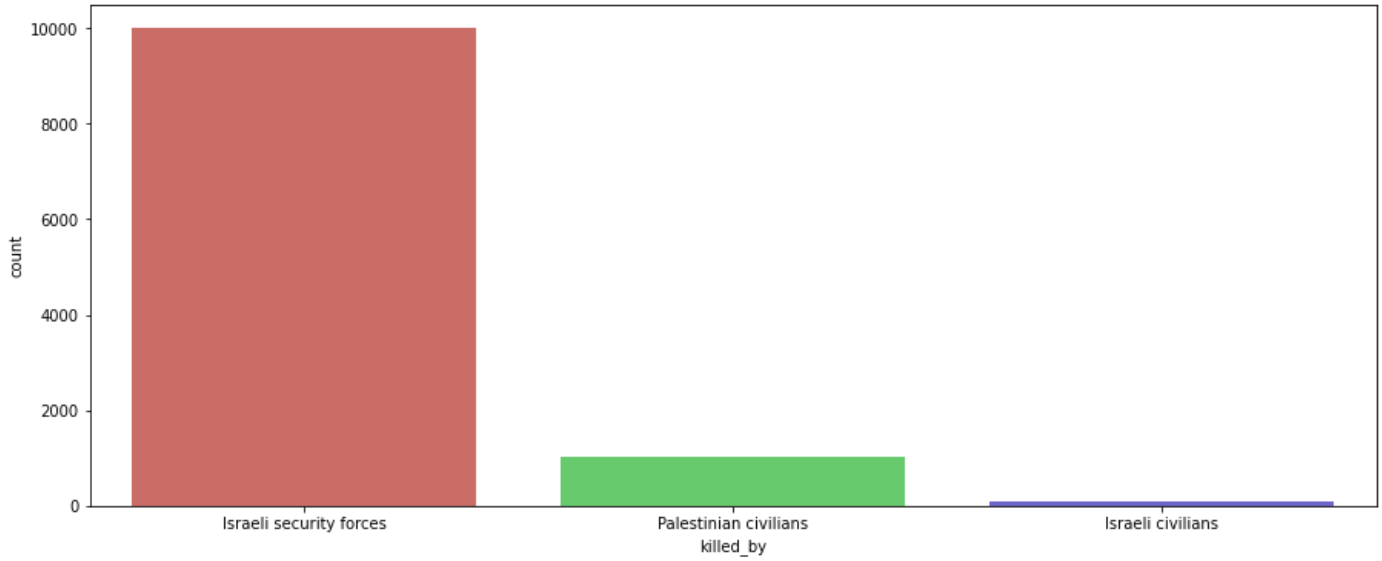
In [24]:
```
for i in categorical:
    plt.figure(figsize=(15,6))
    sns.countplot(df[i], data = df, palette = 'hls')
    plt.show()
```




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

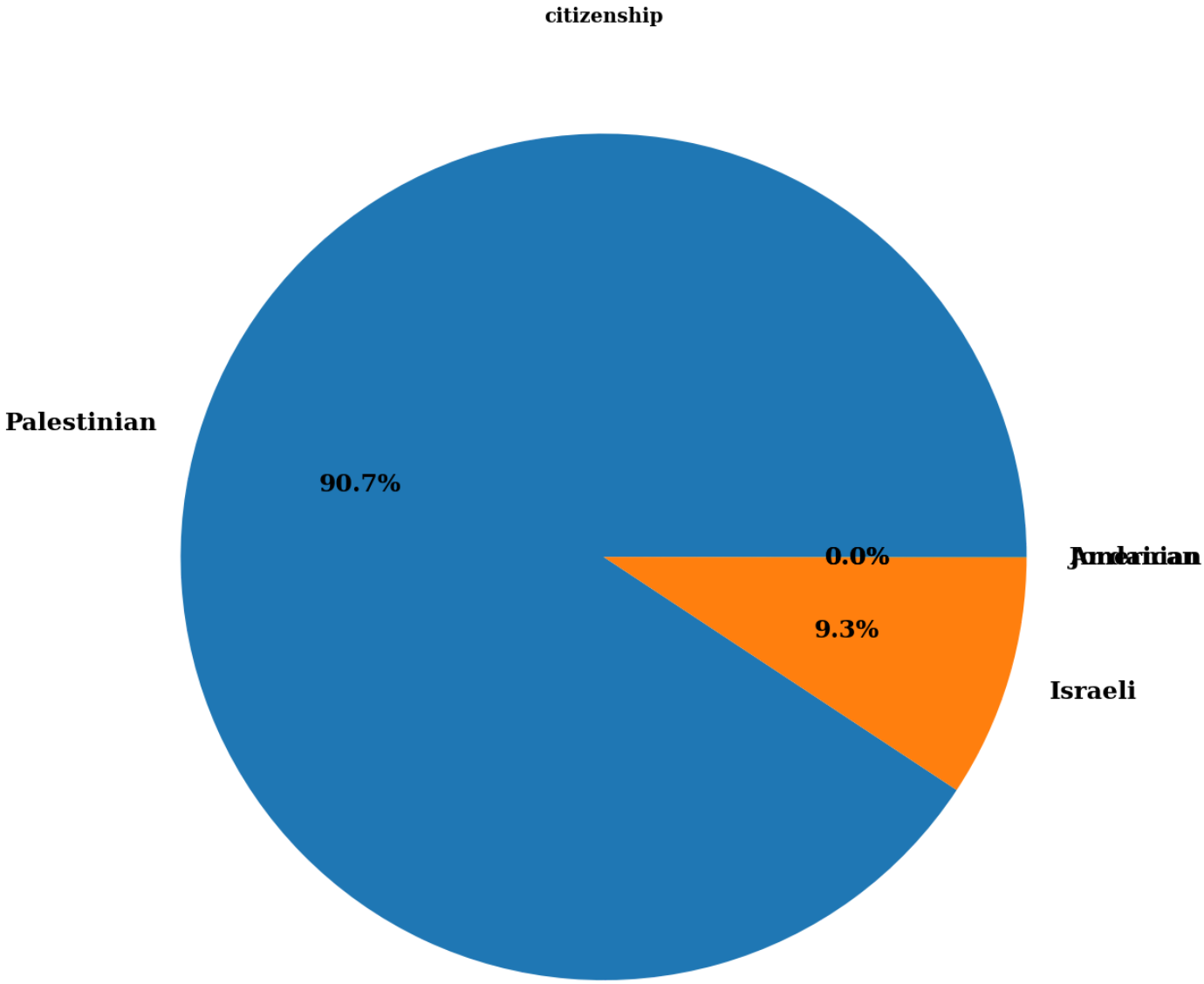

10000
8000
6000
4000
2000
0
count
M
F
Unknown
gender

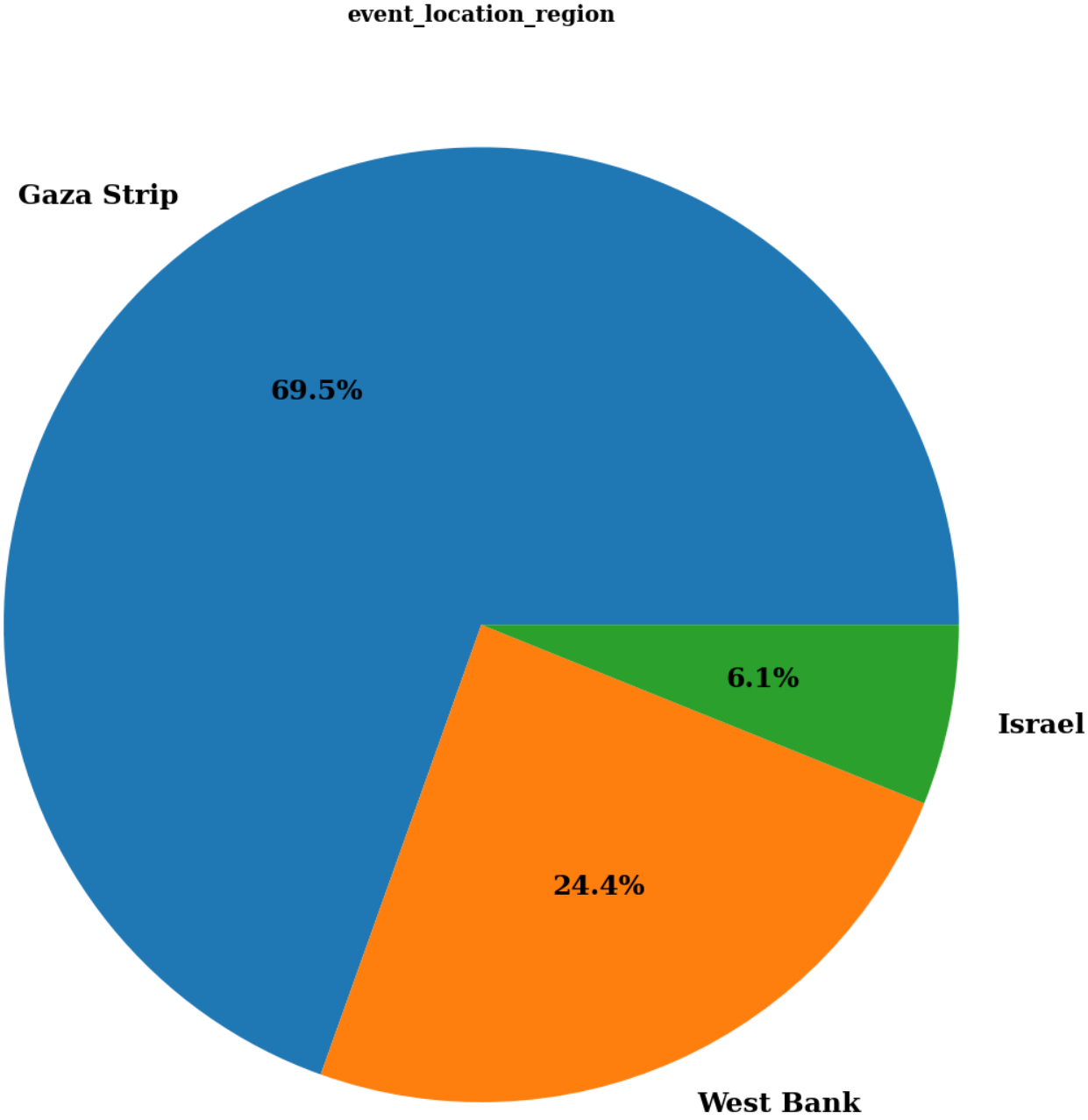

4000
3000
2000
1000
0
count
Not Specified
No
Yes
Unknown
Israelis
Object of targeted killing
took_part_in_the_hostilities

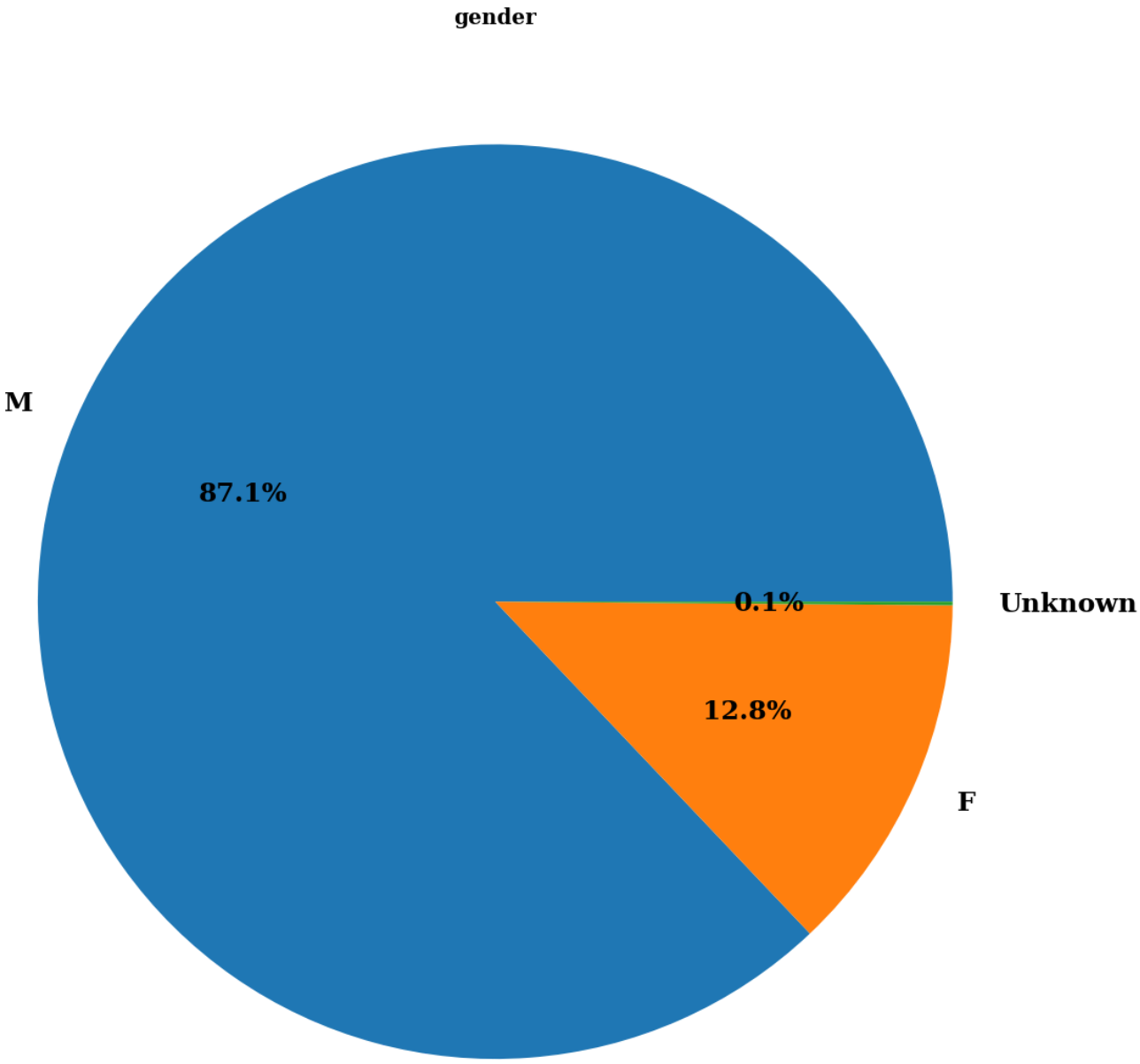


In [25]:

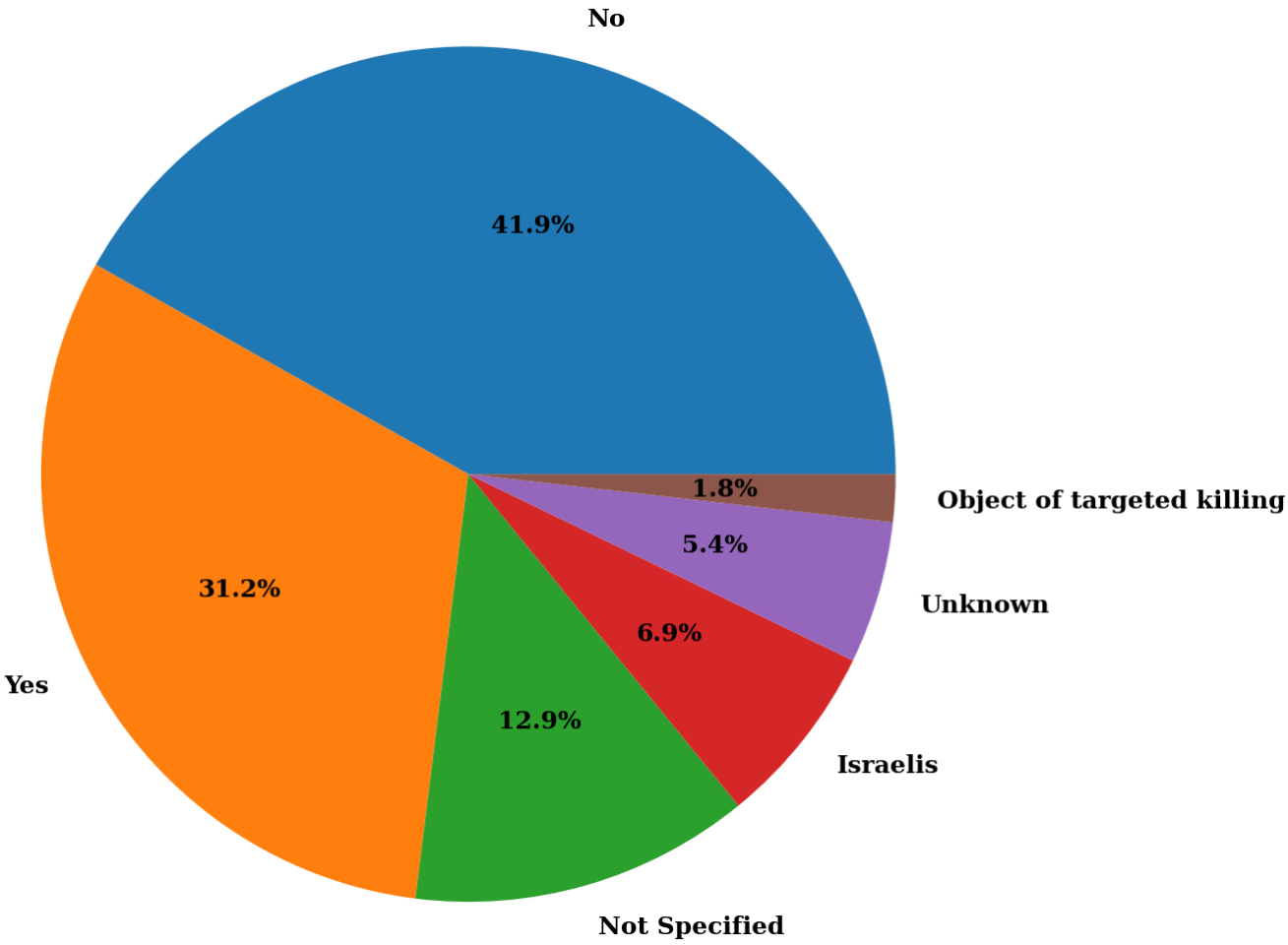
```
for i in categorical:
    plt.figure(figsize=(30,20))
    plt.pie(df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.1f%%',
                                            'color': 'black',
                                            'weight': 'bold',
                                            'family': 'serif' })
    hfont = {'fontname':'serif', 'weight': 'bold'}
    plt.title(i, size=20, **hfont)
    plt.show()
```

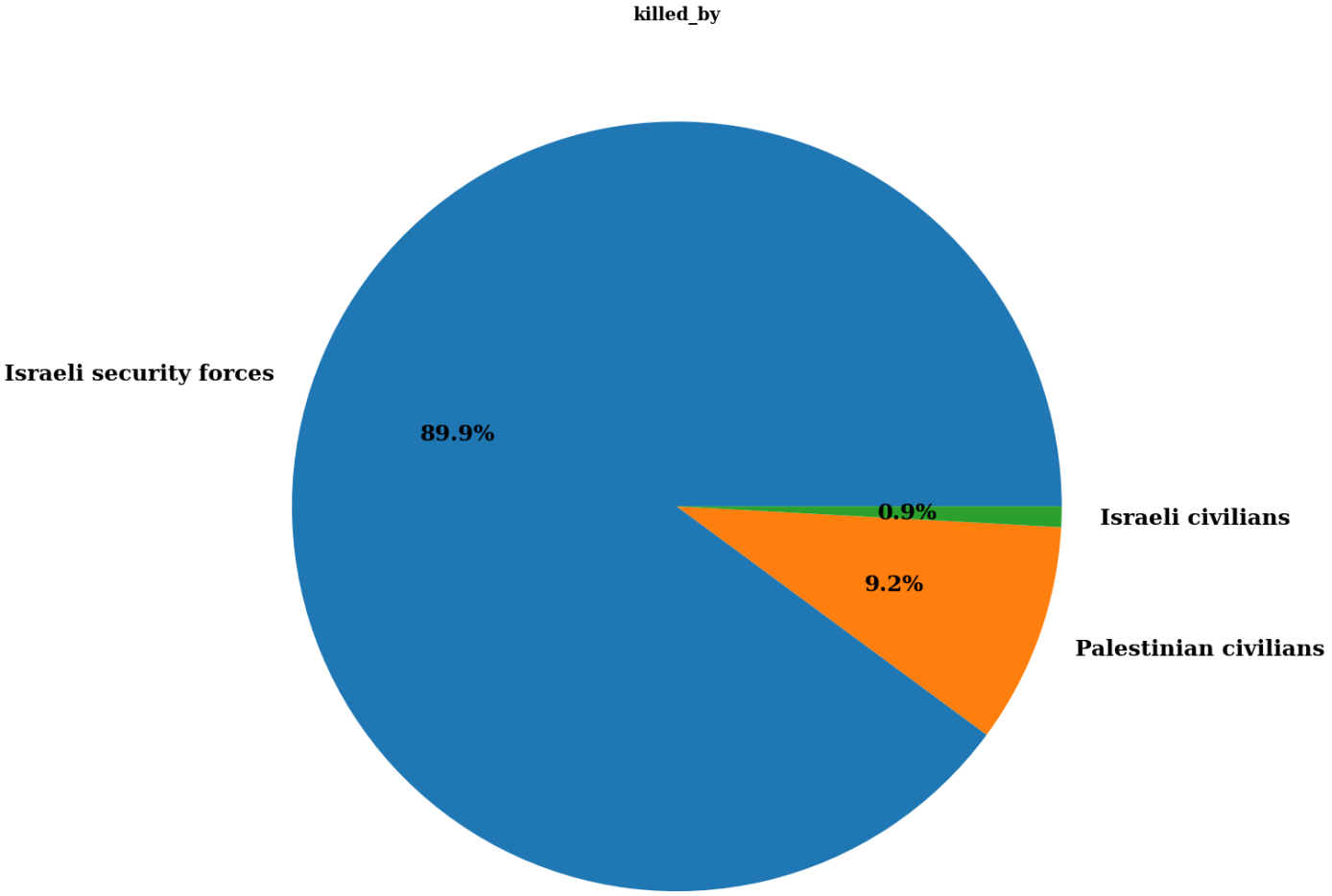

citizenship
Palestinian
90.7%
0.0%
9.3%
Israeli


event_location_region
Gaza Strip
69.5%
6.1%
Israel
24.4%
West Bank

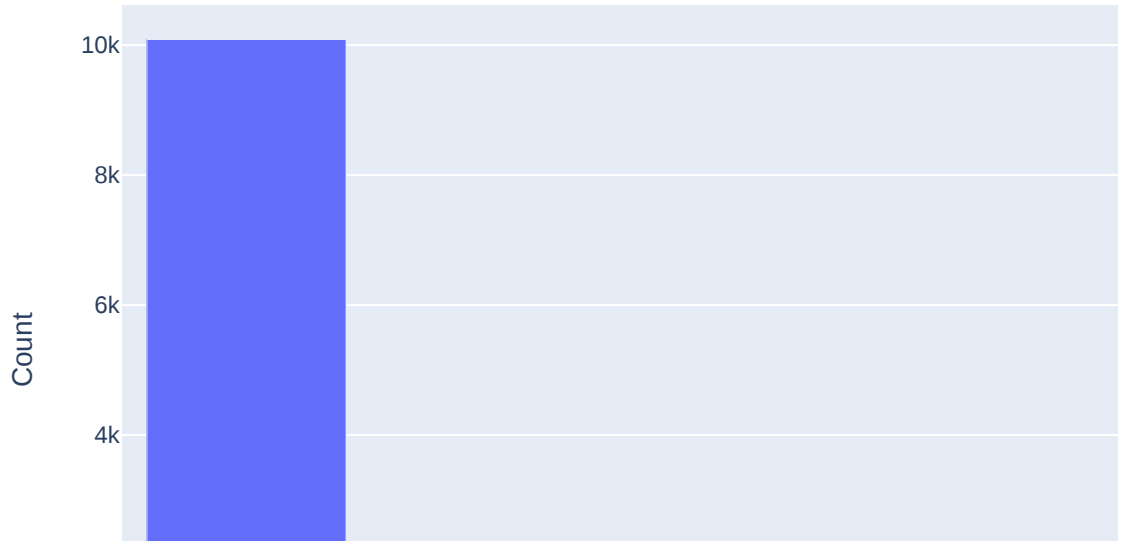

gender
M
87.1%
0.1%
Unknown
12.8%
F

took_part_in_the_hostilities

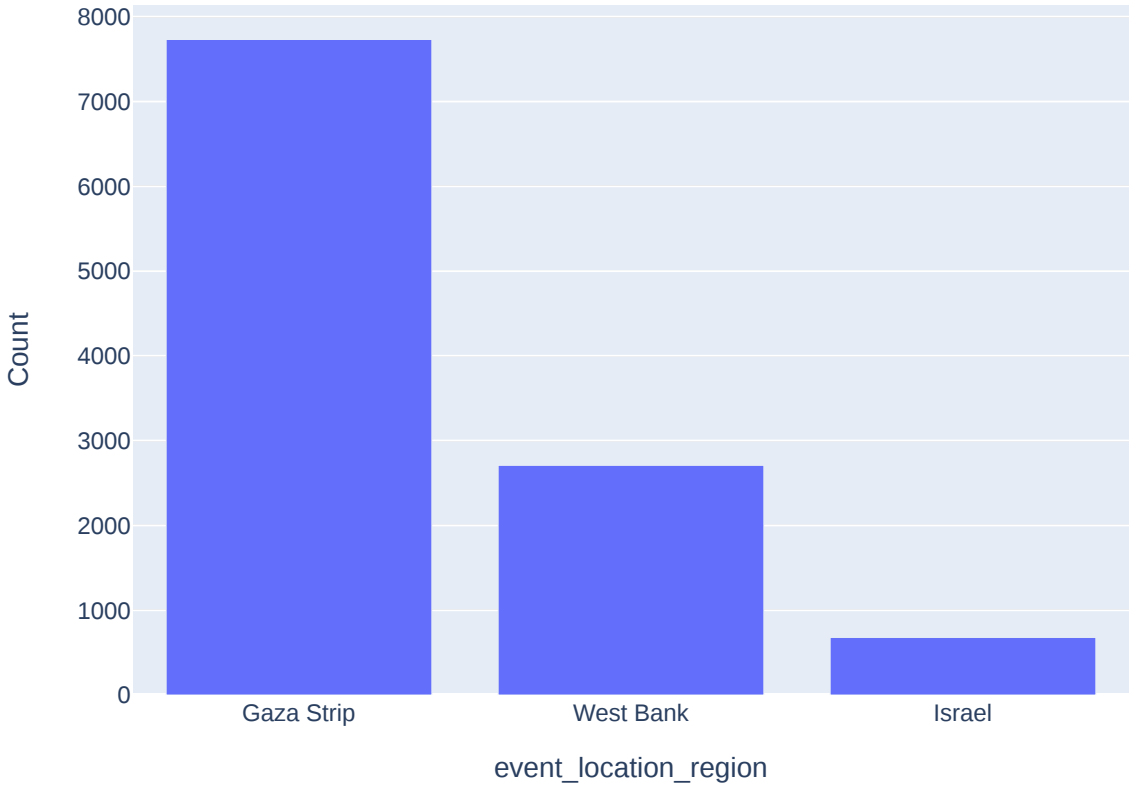




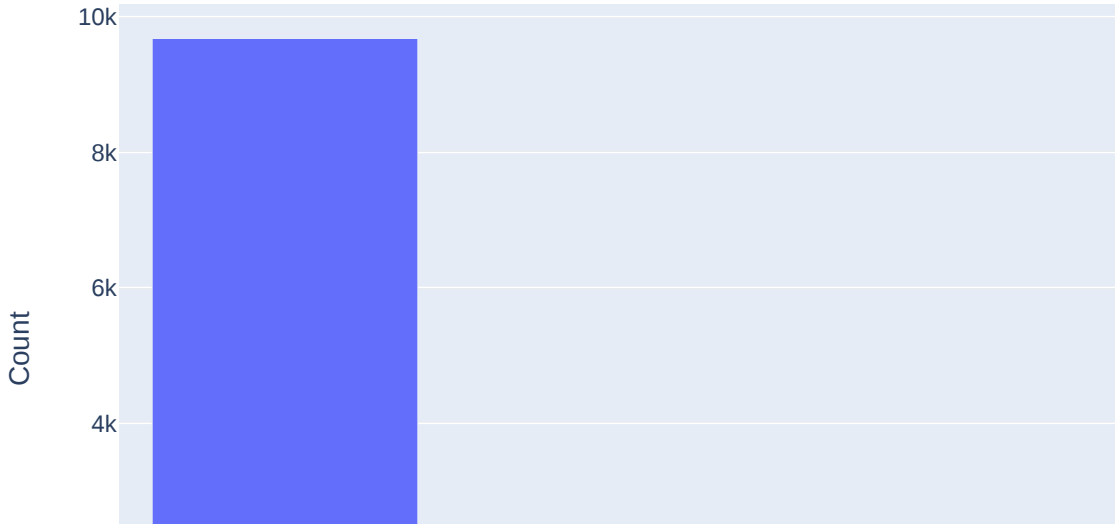
In [26]:
```
for i in categorical:
    fig = go.Figure(data=[go.Bar(x=df[i].value_counts().index,
                                 y=df[i].value_counts())])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Count")
    fig.show()
```

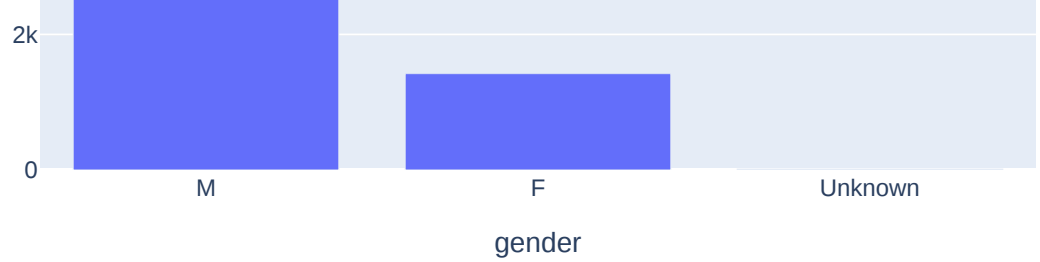


event_location_region

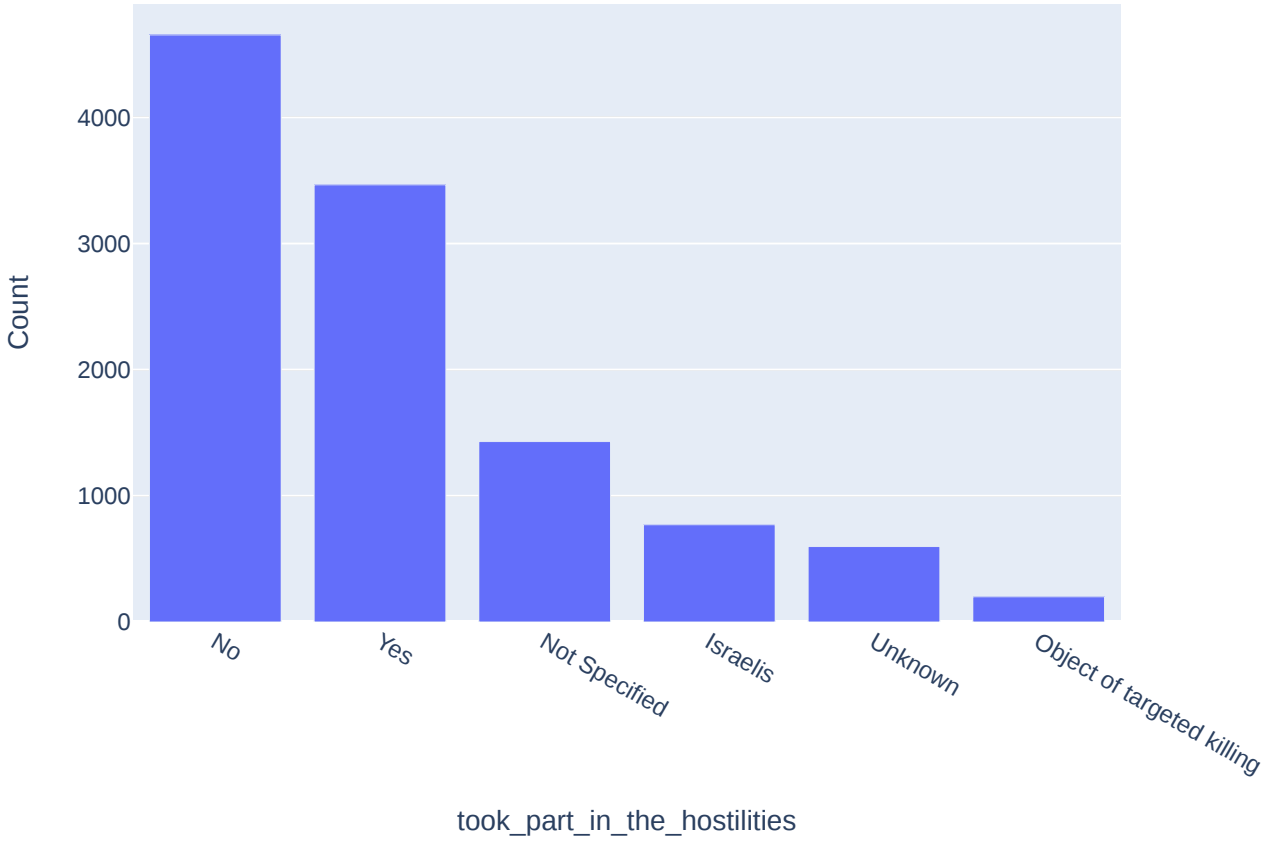


gender

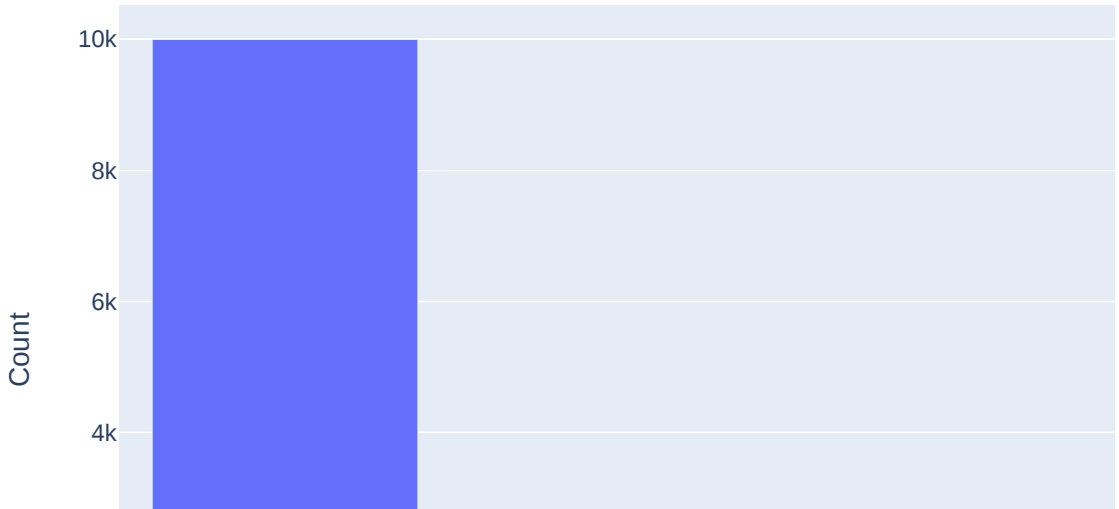


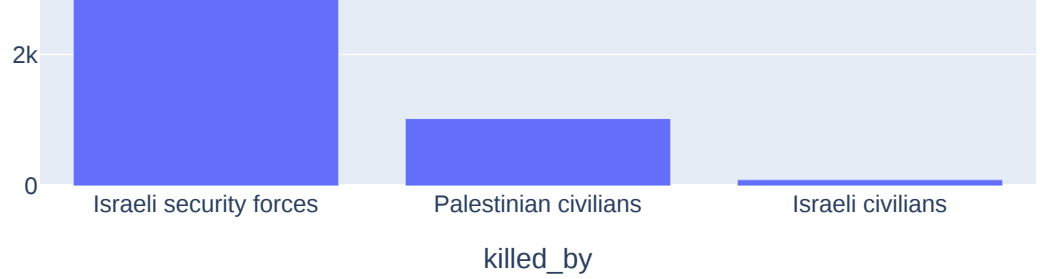


took_part_in_the_hostilities



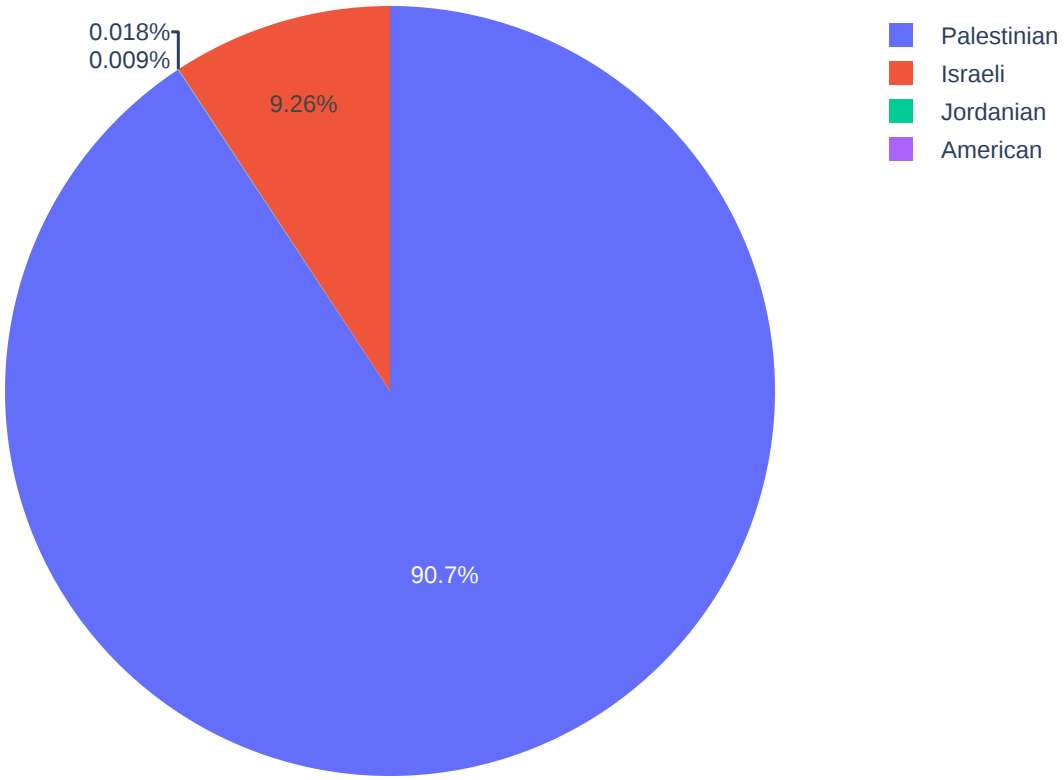
killed_by



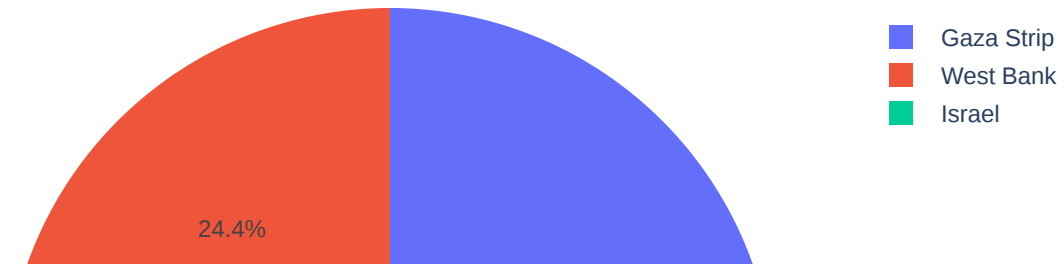


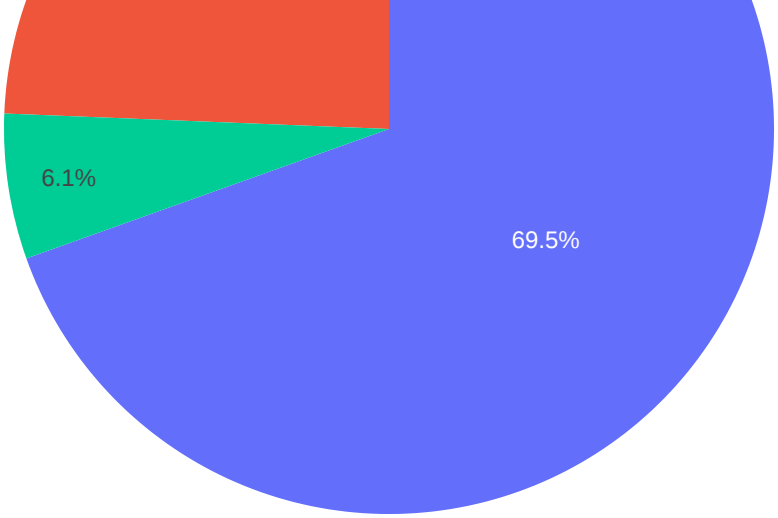
In [27]:
```
for i in categorical:
    print('Pie plot for:', i)
    fig = px.pie(df, names=i)
    fig.show()
    print('\n')
```

```
Pie plot for: citizenship
```

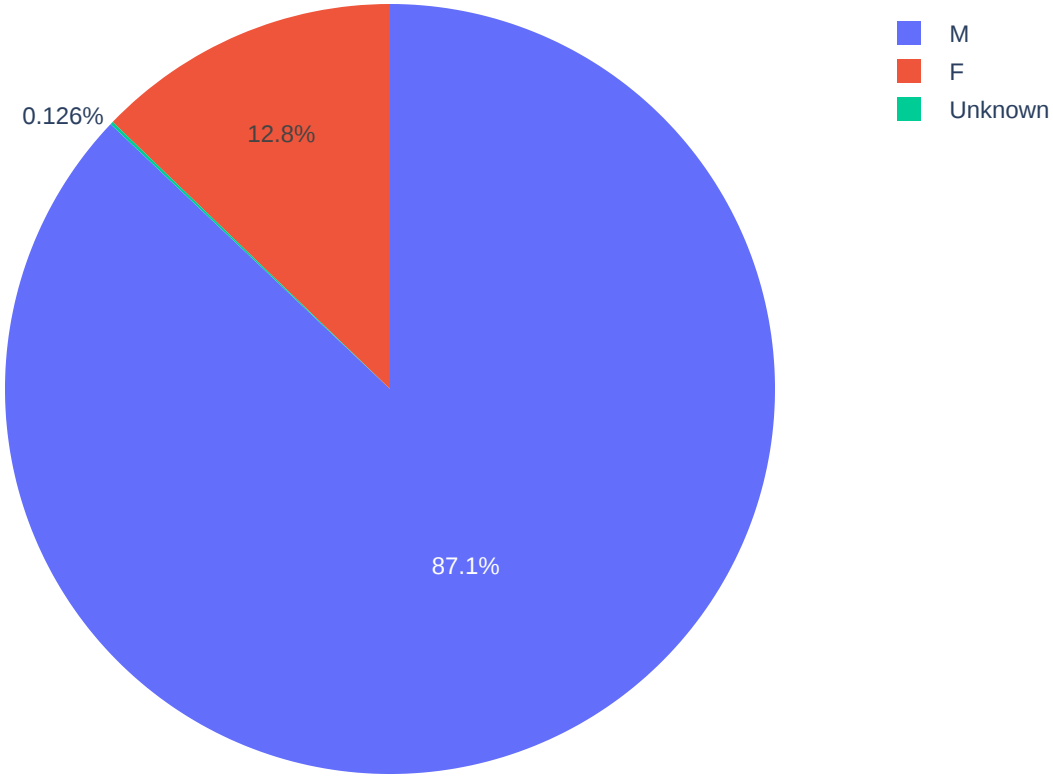



```
Pie plot for: event_location_region
```

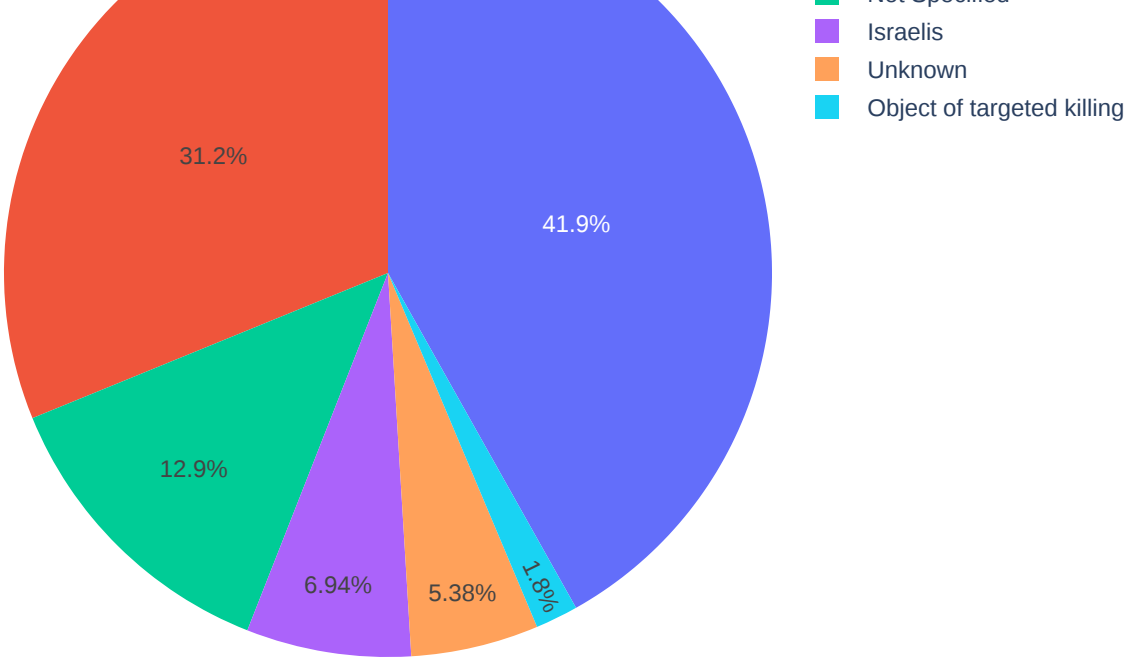




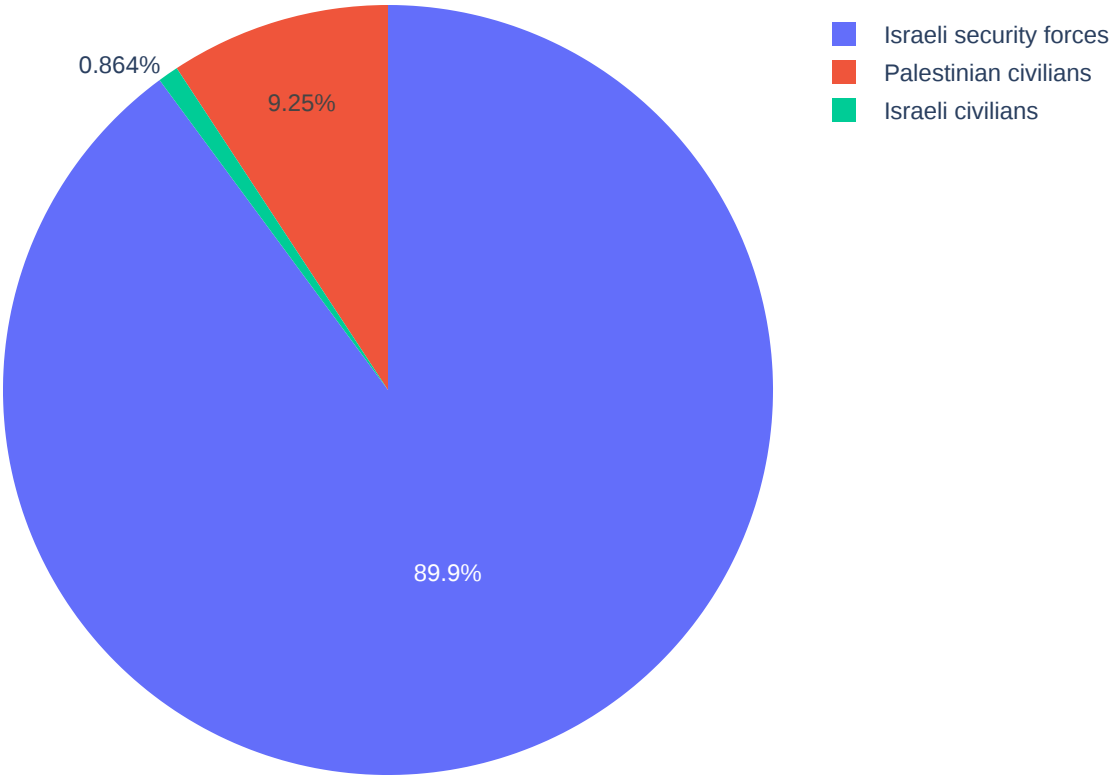
Pie plot for: gender



Pie plot for: took_part_in_the_hostilities

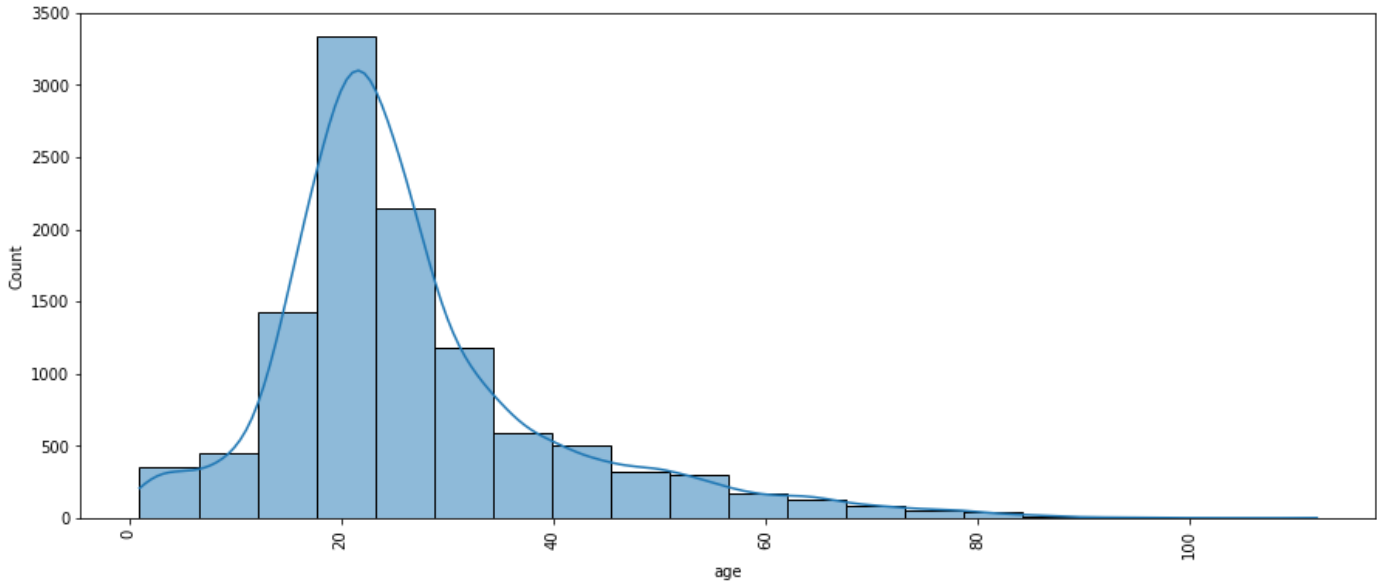


Pie plot for: killed_by

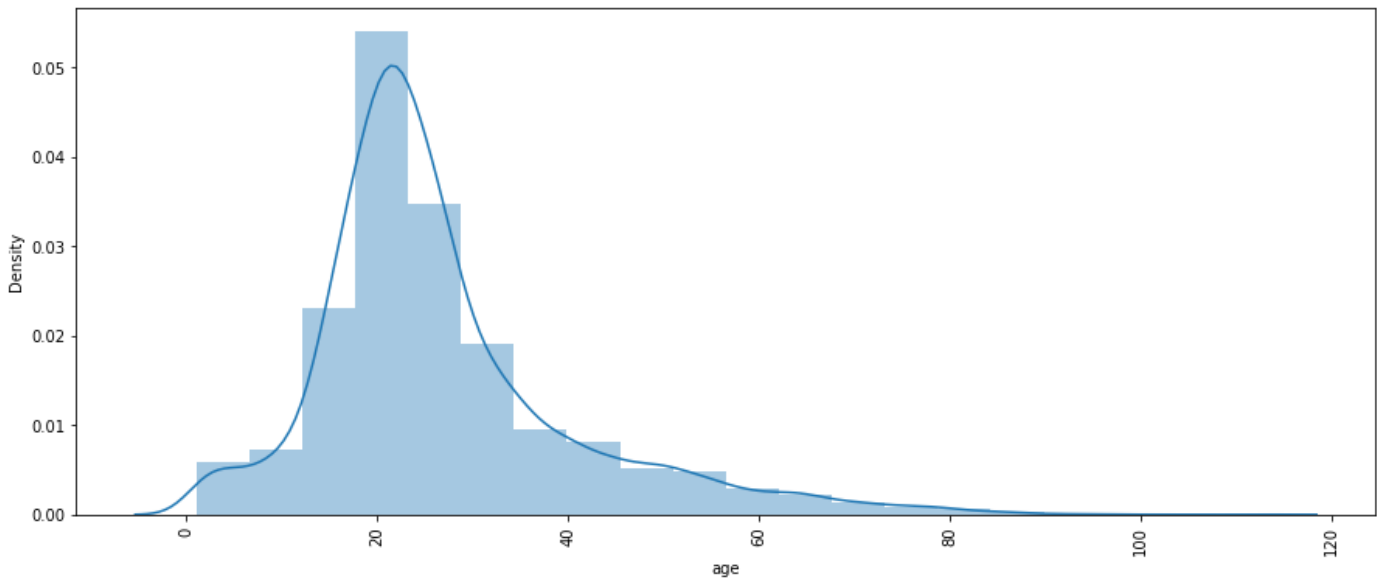


In [28]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
```

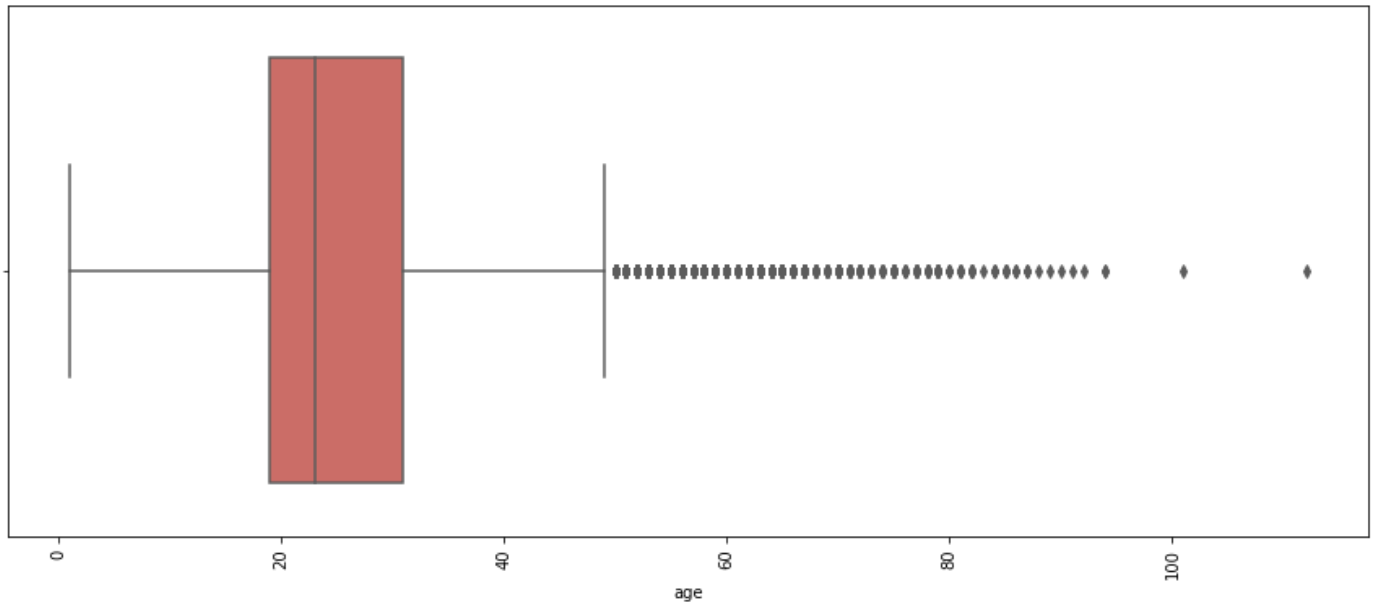
```
    sns.histplot(df[i], kde = True, bins = 20, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



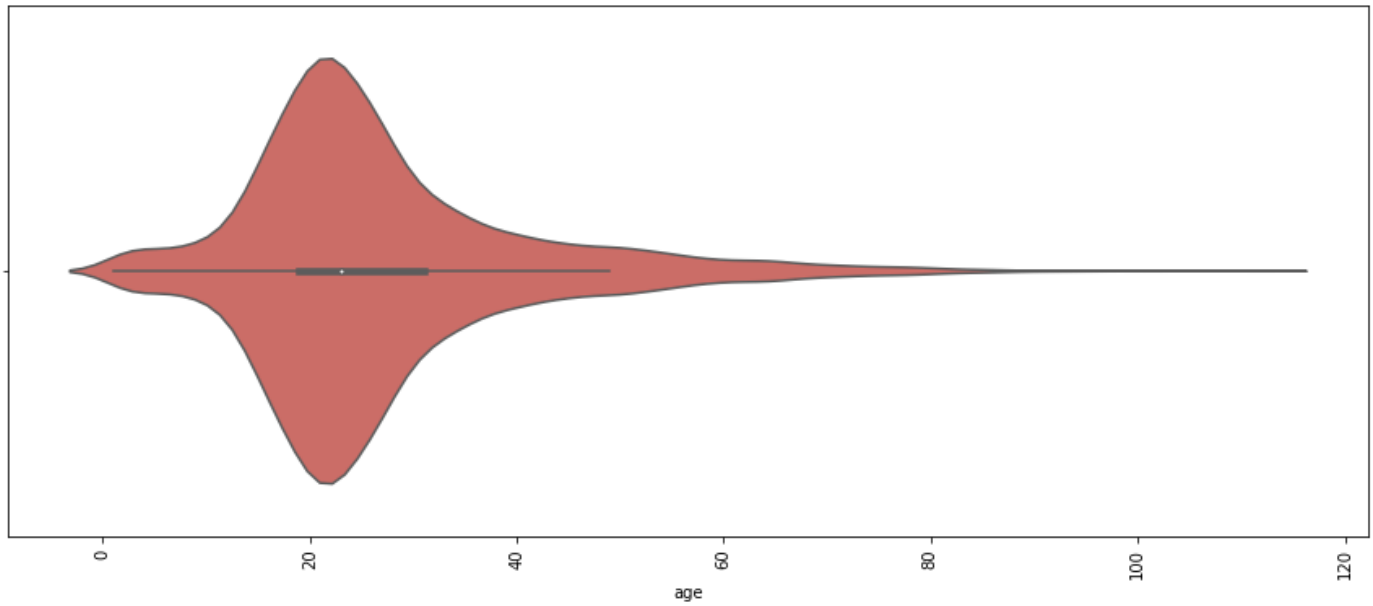
In [29]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.distplot(df[i], kde = True, bins = 20)
    plt.xticks(rotation = 90)
    plt.show()
```



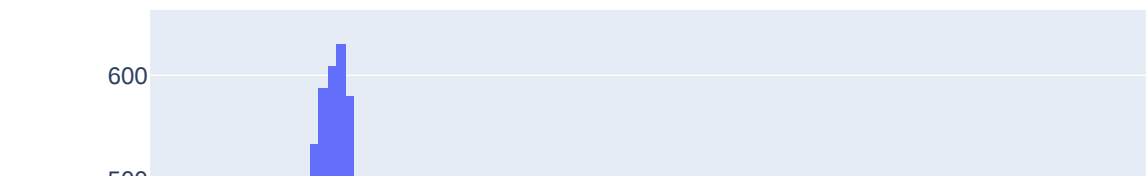
In [30]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.boxplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

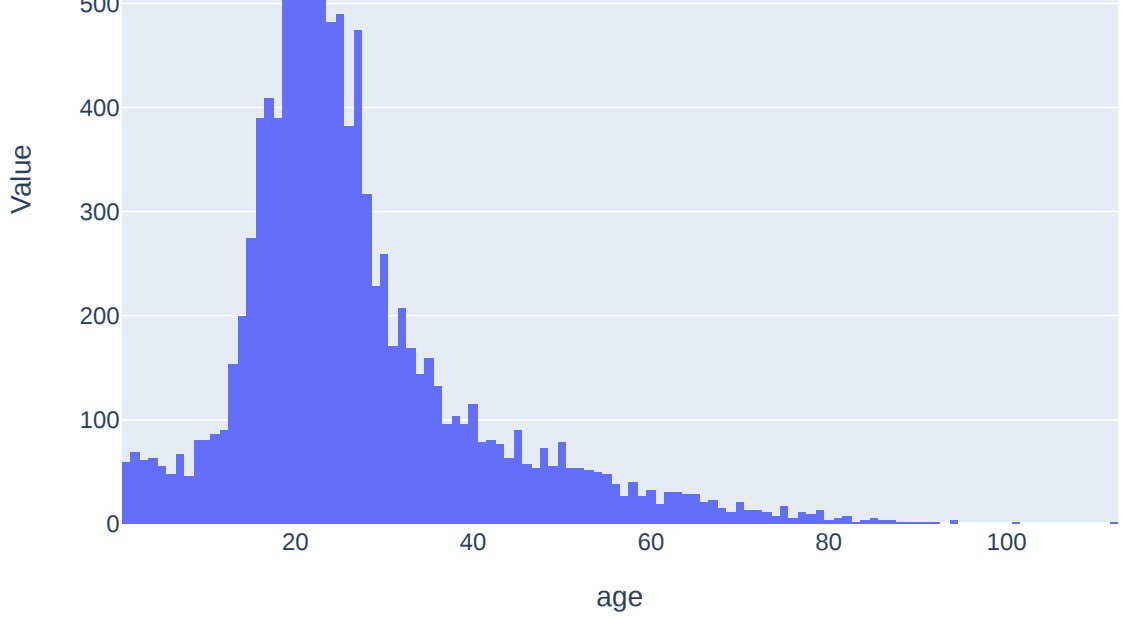


In [31]:
```
for i in continuous:
    plt.figure(figsize=(15,6))
    sns.violinplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

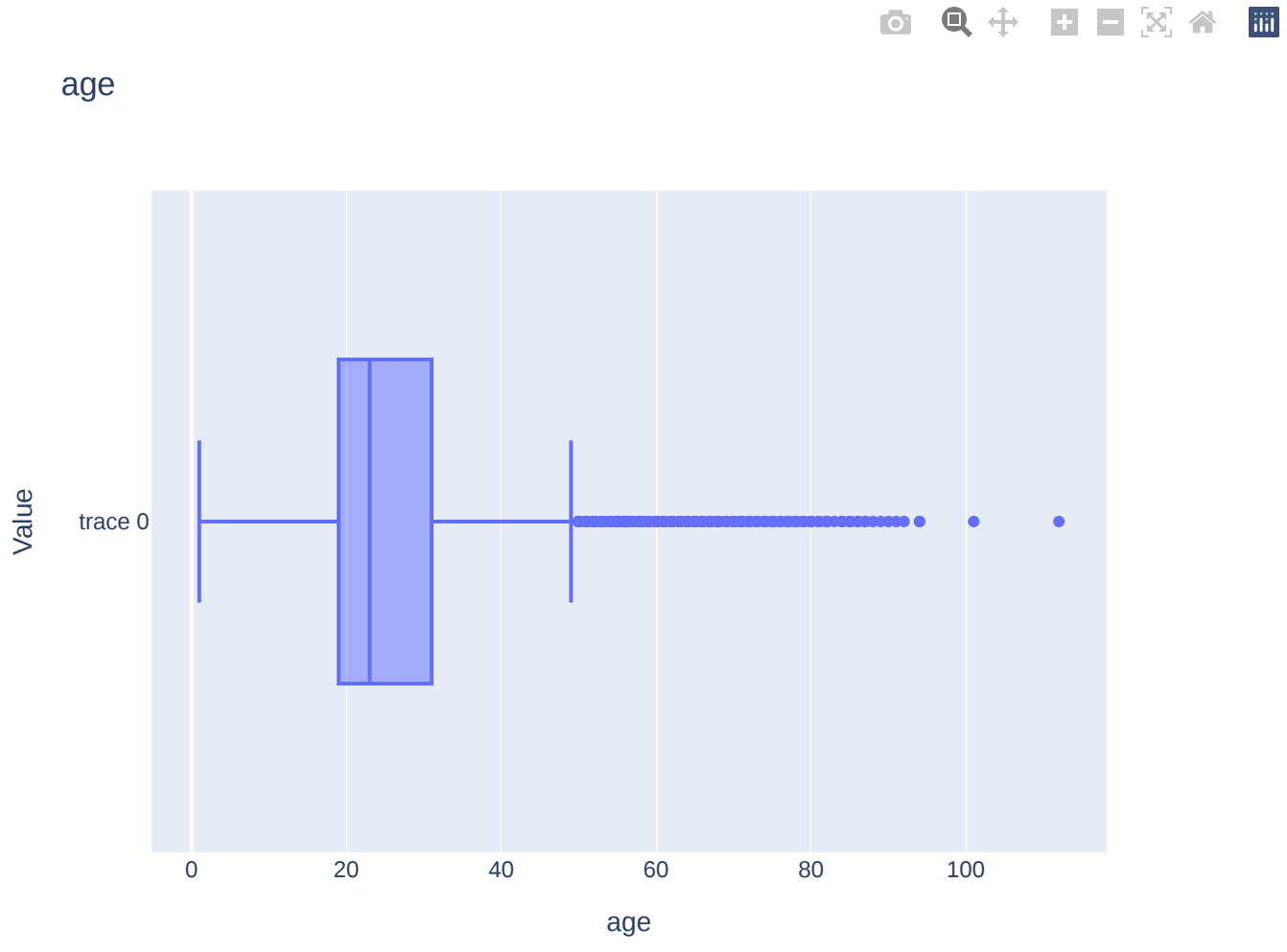


In [32]:
```
for i in numerical_columns:
    fig = go.Figure(data=[go.Histogram(x=df[i])])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value")
    fig.show()
```



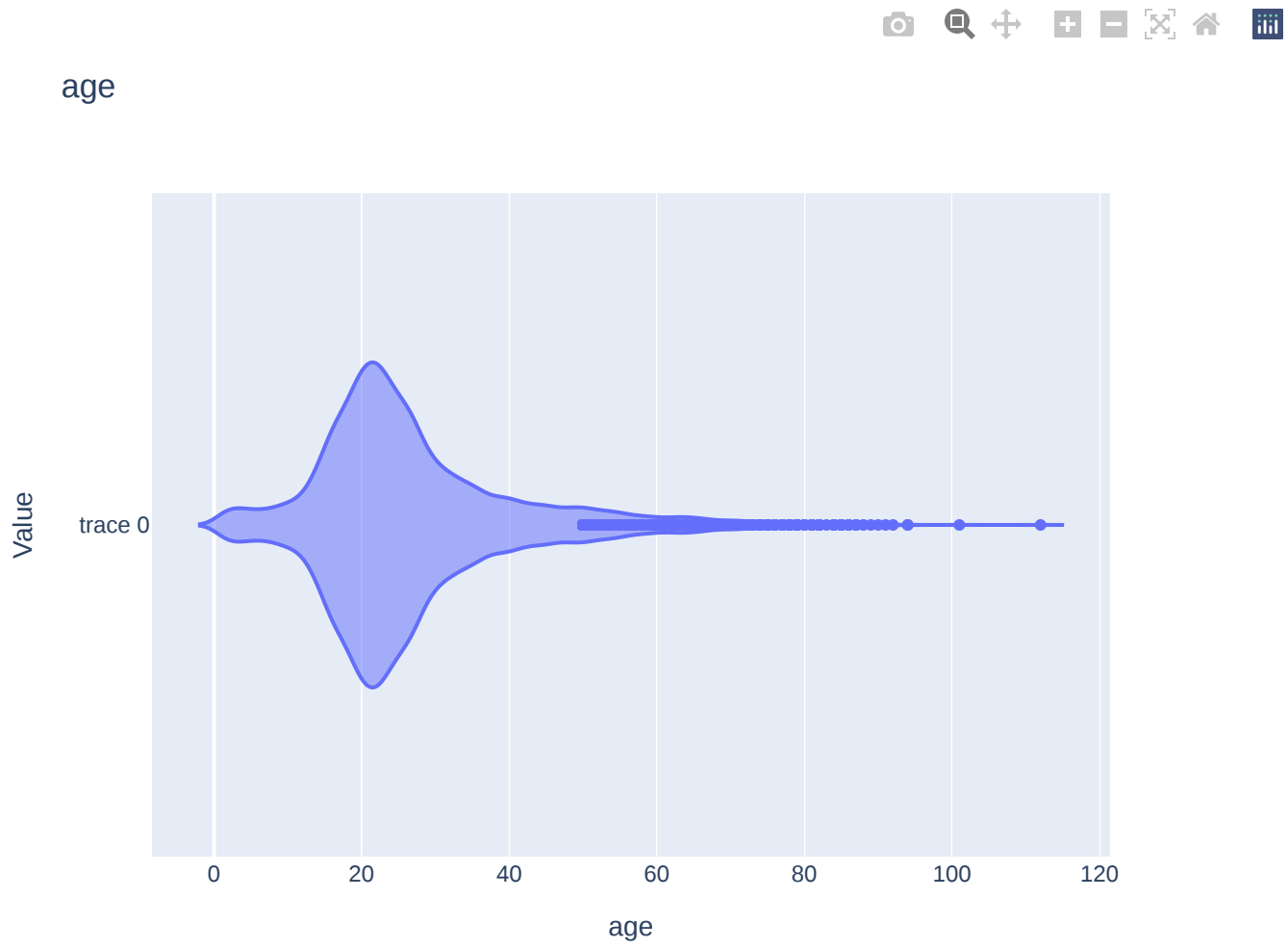


In [33]:
```
for i in numerical_columns:
    fig = go.Figure(data=[go.Box(x=df[i])])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value")
    fig.show()
```



In [34]:
```
for i in numerical_columns:
```

```
    fig = go.Figure(data=[go.Violin(x=df[i])])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value")
    fig.show()
```

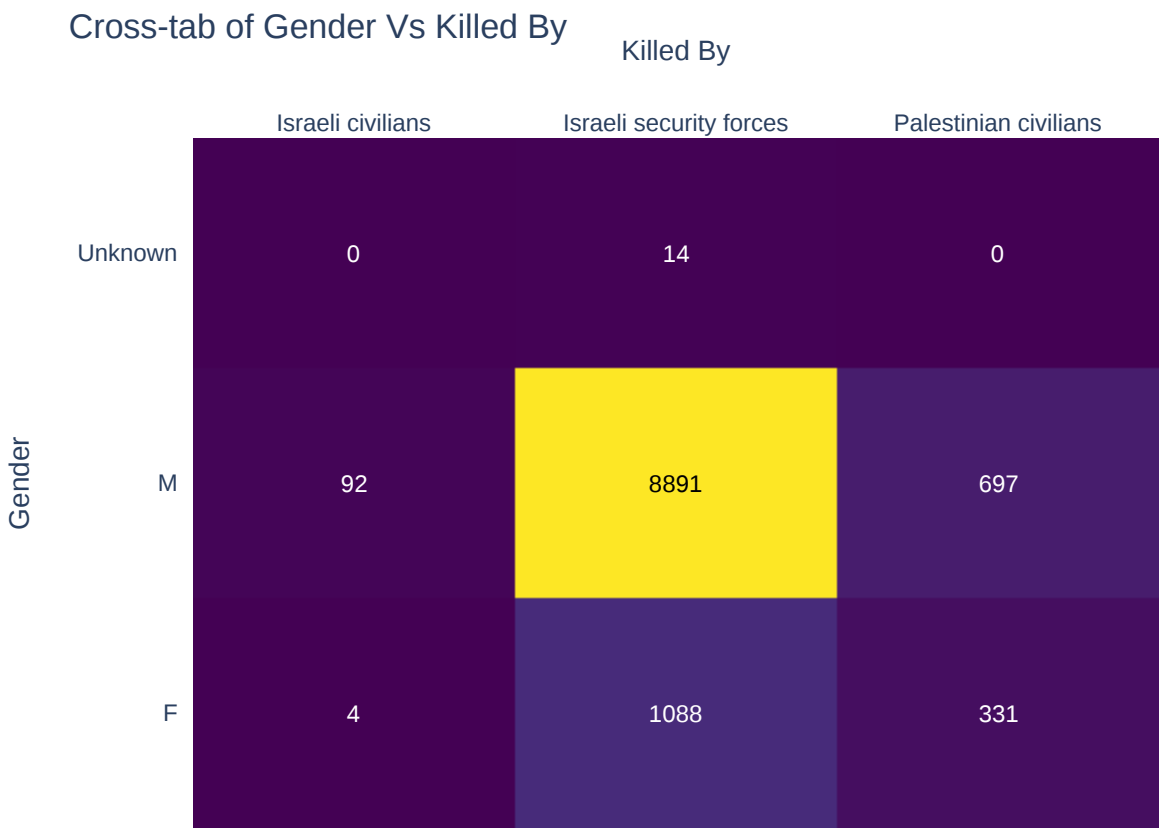


In [35]:
```
cross_tab = pd.crosstab(df['gender'], df['killed_by'])
print('Cross-tabulation of gender and killed_by:')
print(cross_tab)
```

```
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
```

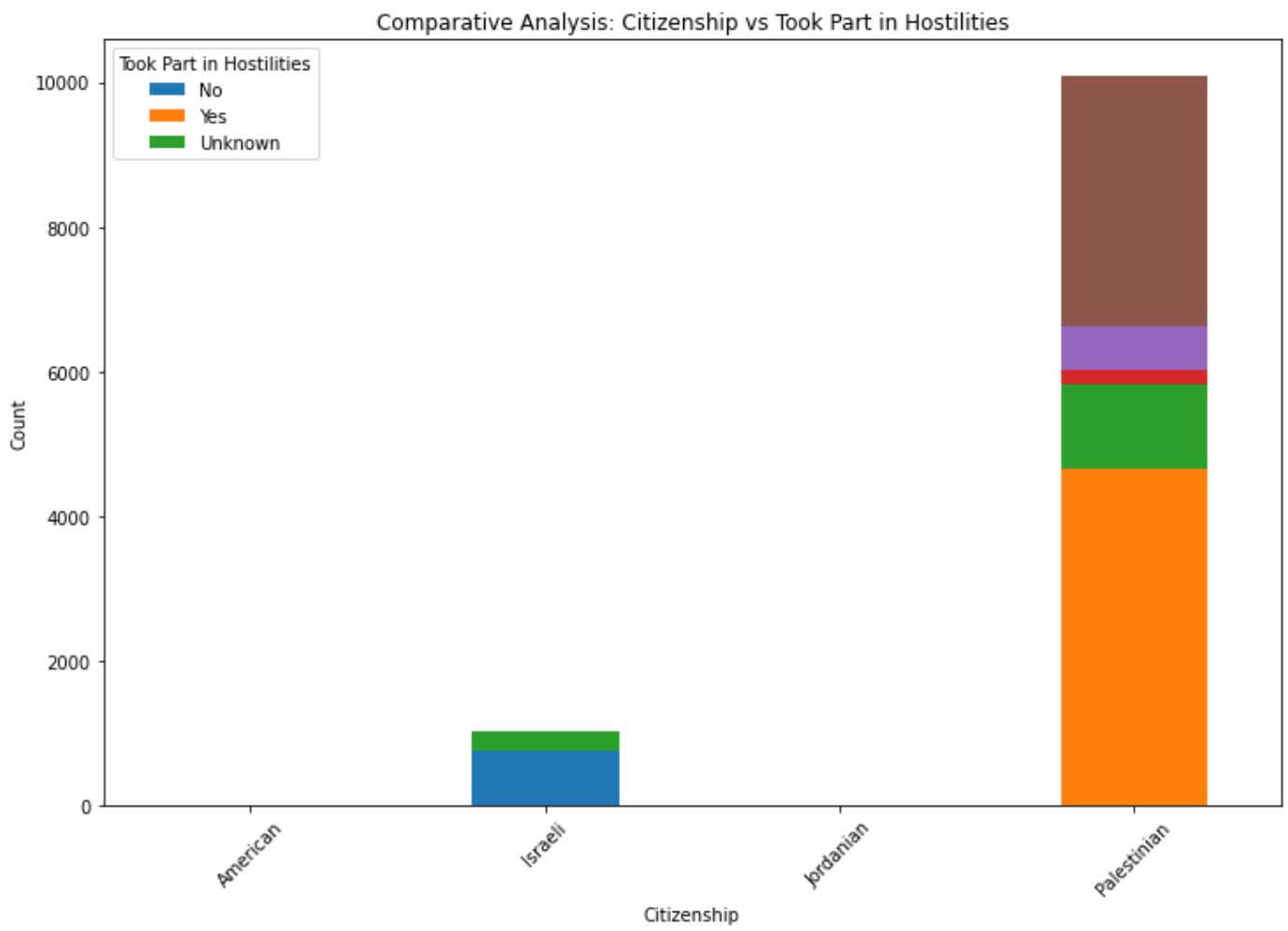
In [36]:
```
import plotly.figure_factory as ff

cross_tab = pd.crosstab(df['gender'], df['killed_by'])

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()
```



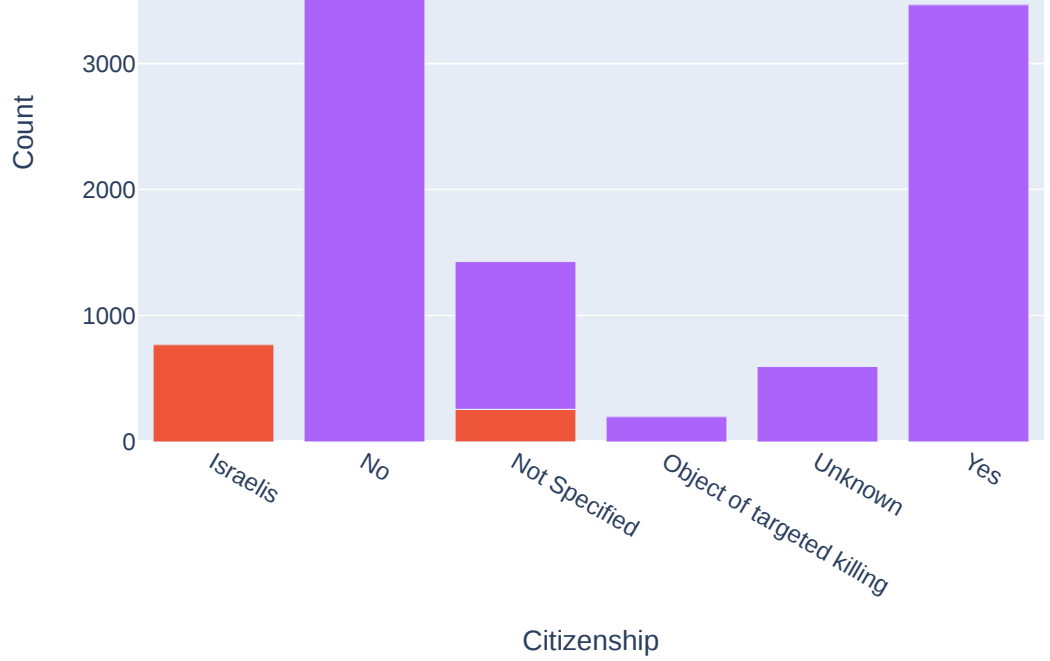
In [37]:

```
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).size().un
    grouped_data.plot(kind='bar', stacked=True, figsize=(12, 8))
    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)
```



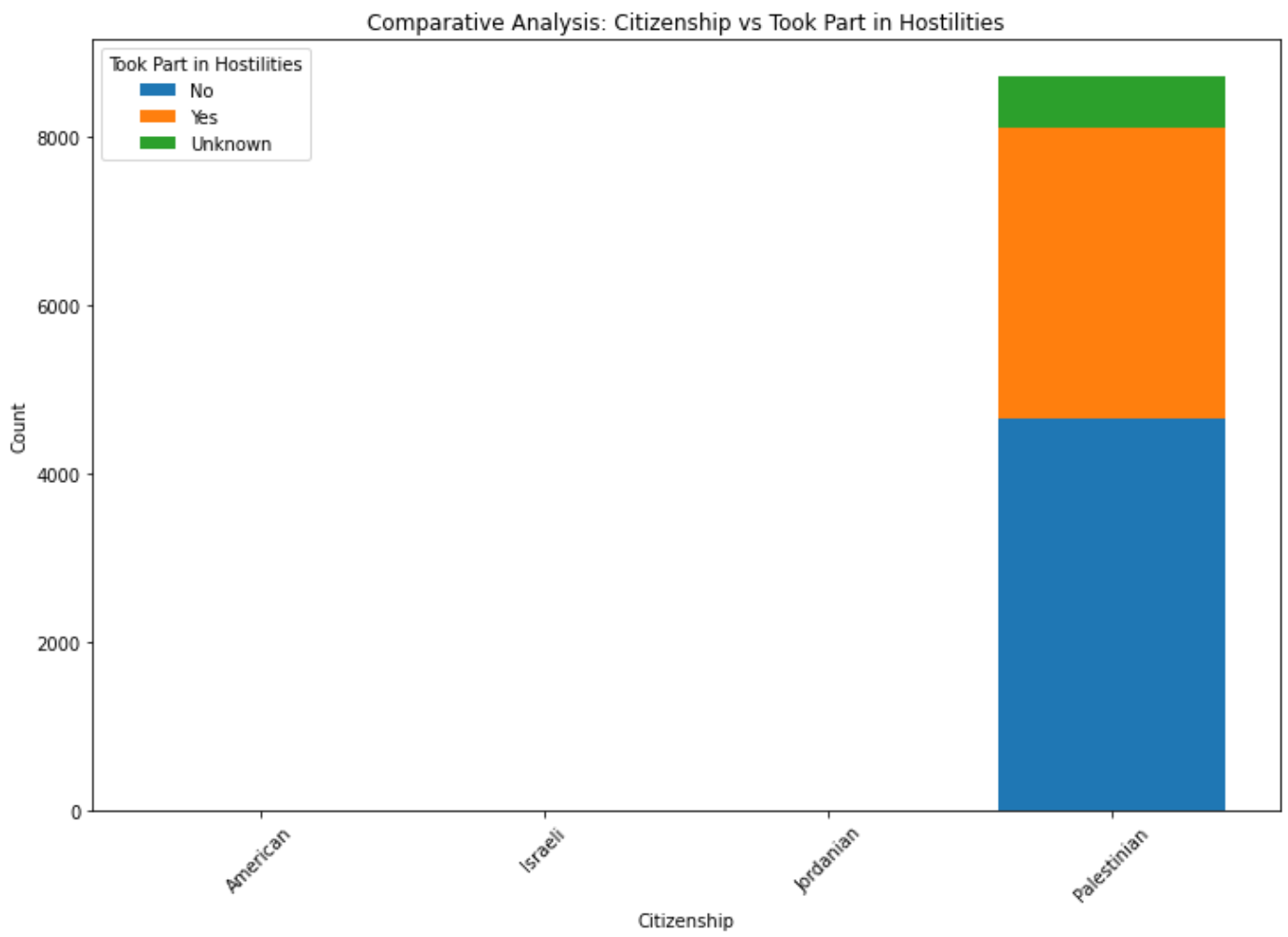
In [38]:

```python
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).size().un

    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 Part in Hostilities',
        xaxis=dict(title='Citizenship'),
        yaxis=dict(title='Count'),
        barmode='stack'
    )

    fig.show()

comparative_analysis(df)
```



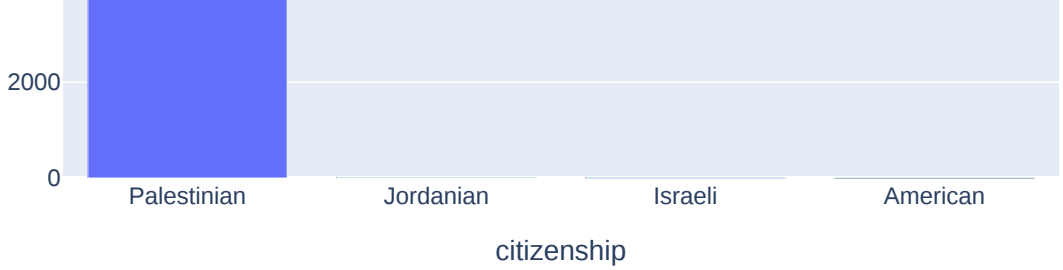
In [39]:

```python
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).size().un

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

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

    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='Y
    plt.bar(grouped_data.index, grouped_data['Unknown'], bottom=grouped_data['No'] + gro
    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()

comparative_analysis(df)
```



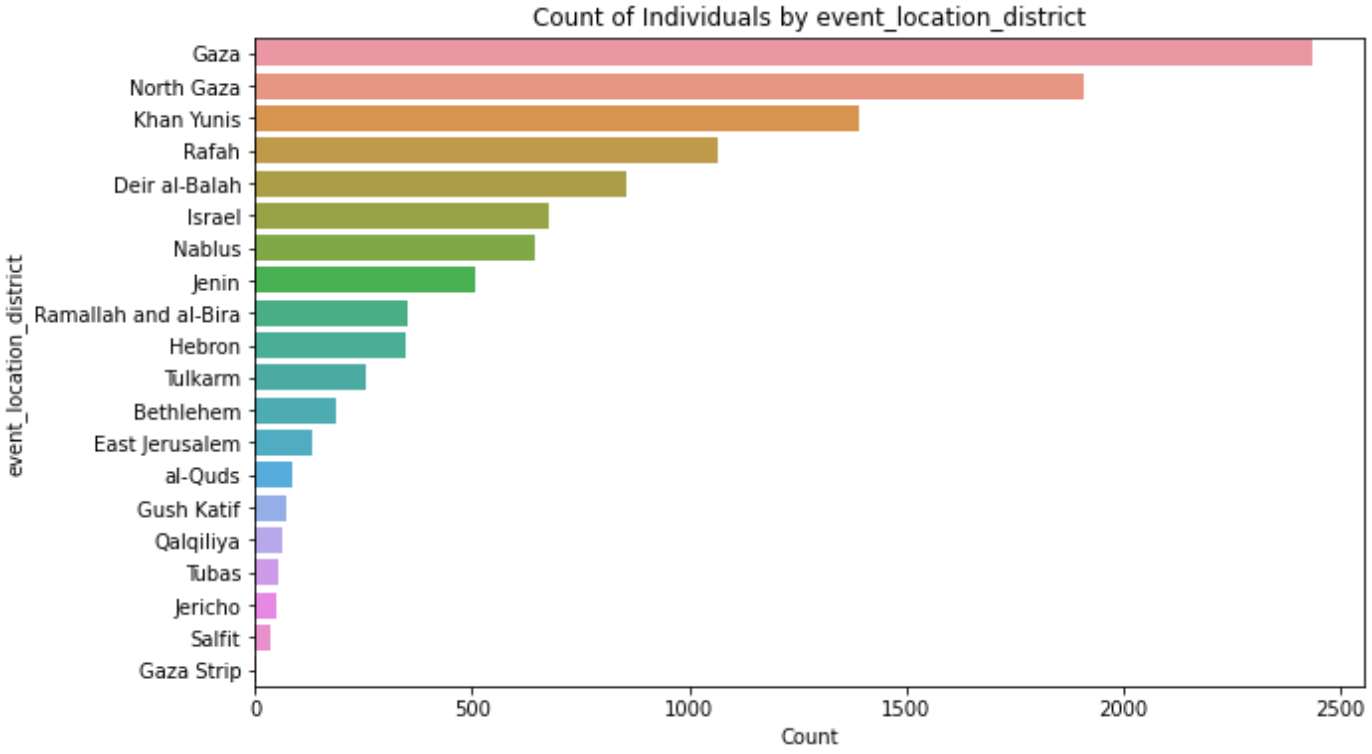
In [40]:

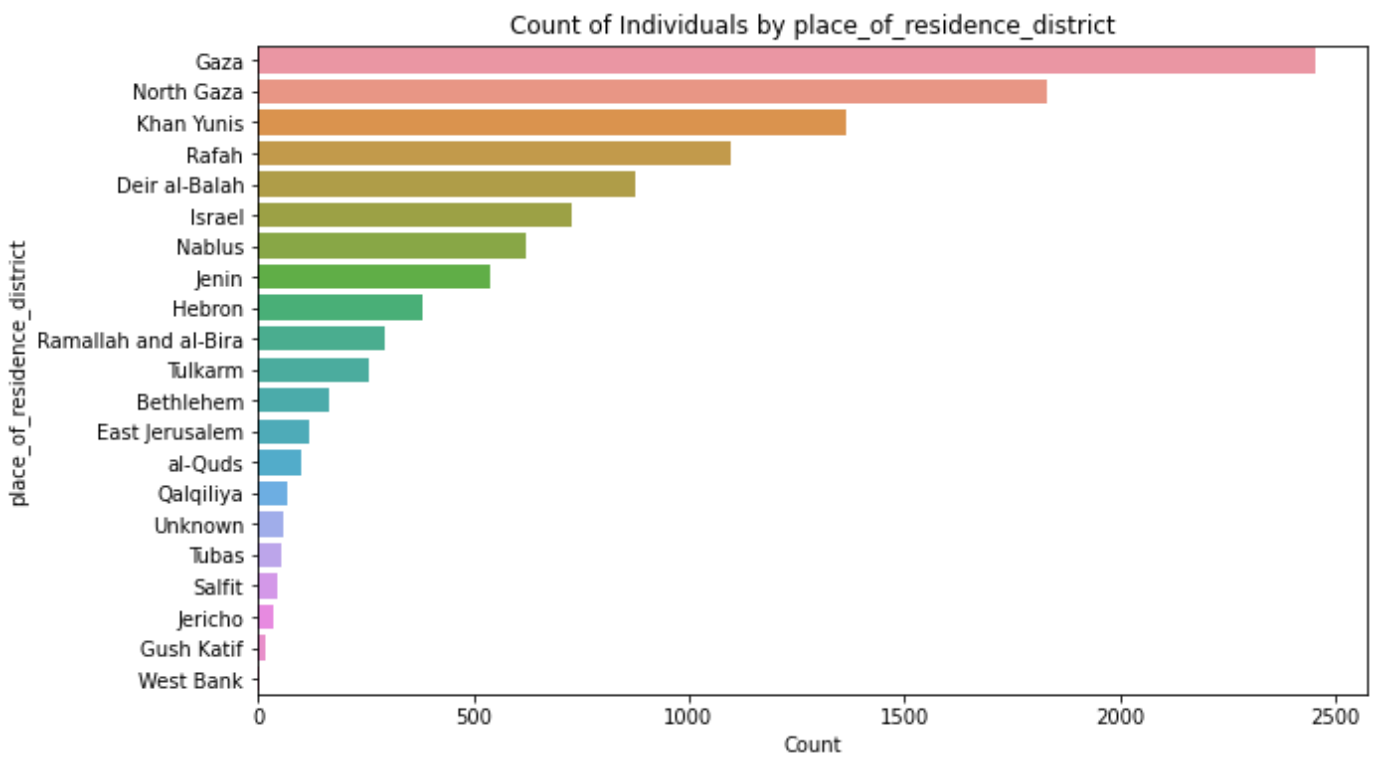
```
def comparative_analysis(df):
    grouped_data = df.groupby(['citizenship', 'took_part_in_the_hostilities']).size().un

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

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

    fig.show()

comparative_analysis(df)
```



In [41]:

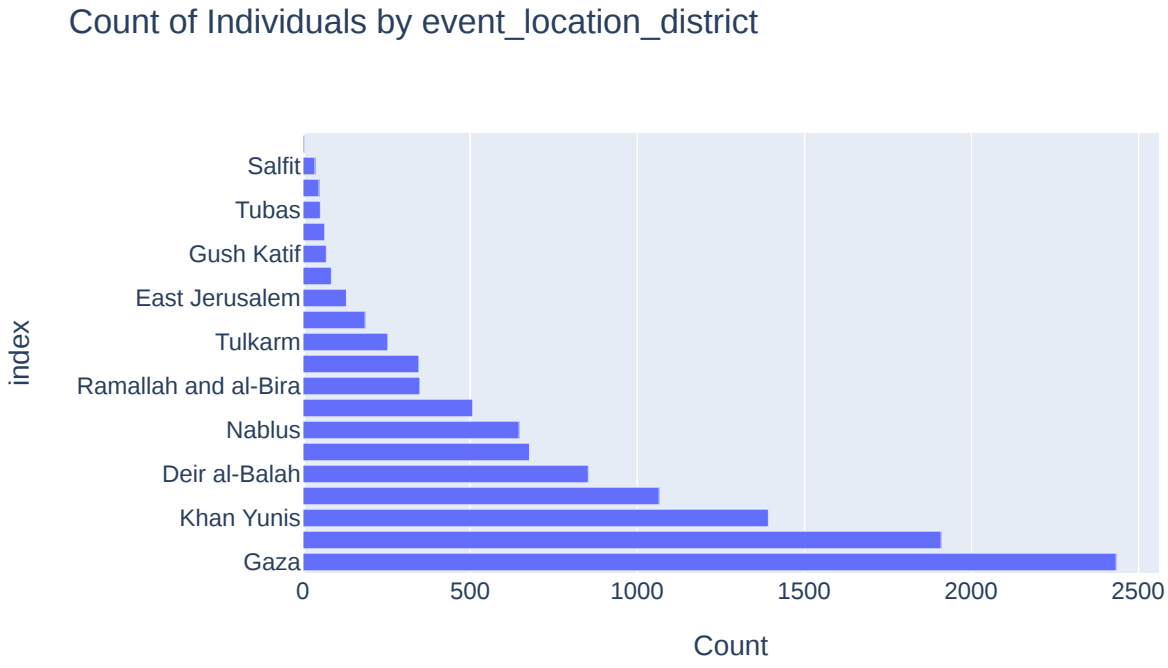
```python
def location_analysis(df):
    location_features = ['event_location_district', 'place_of_residence_district']

    for feature in location_features:
        plt.figure(figsize=(10, 6))
        sns.countplot(data=df, y=feature, order=df[feature].value_counts().index)
        plt.title(f'Count of Individuals by {feature}')
        plt.xlabel('Count')
        plt.ylabel(feature)
        plt.show()

location_analysis(df)
```

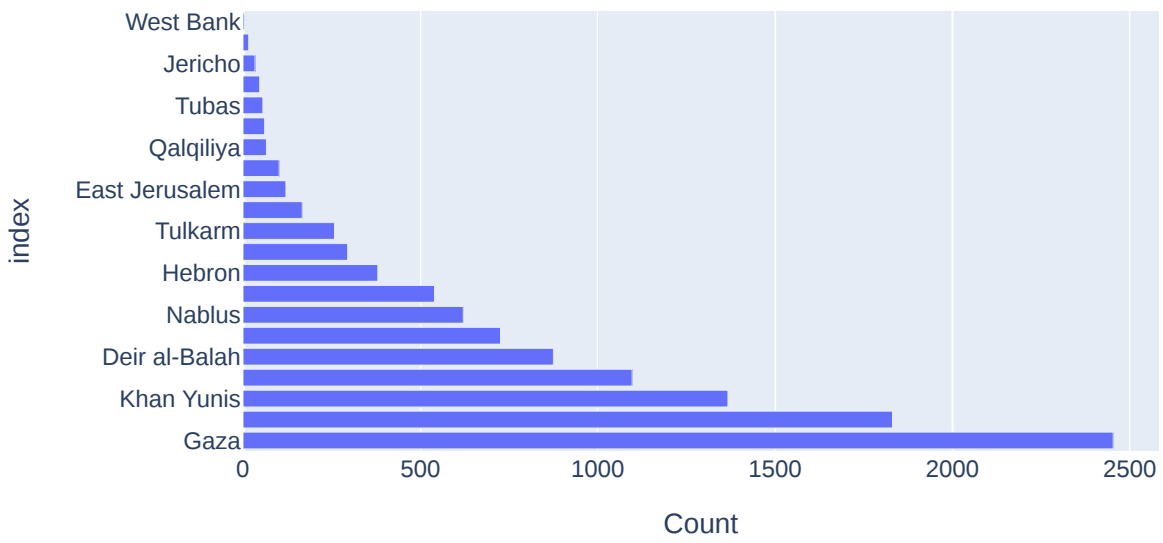




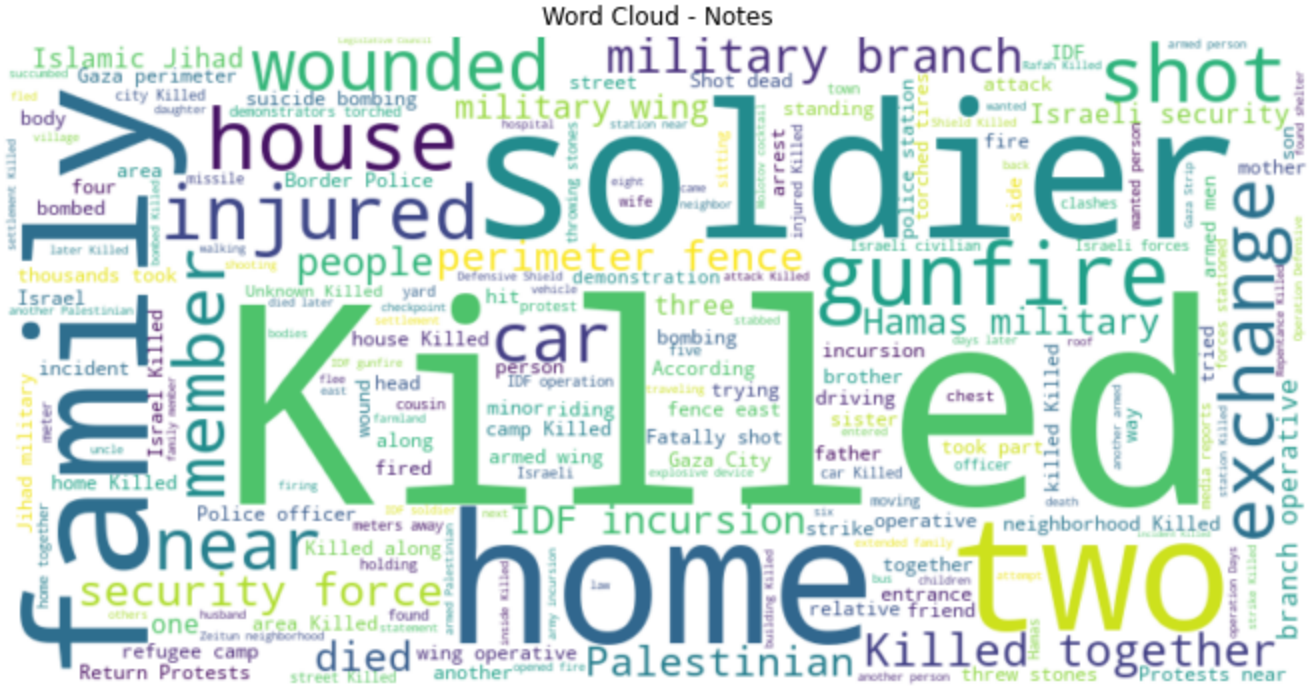
In [42]:

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

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

        fig.show()

location_analysis(df)
```





In [43]:
```
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)
```

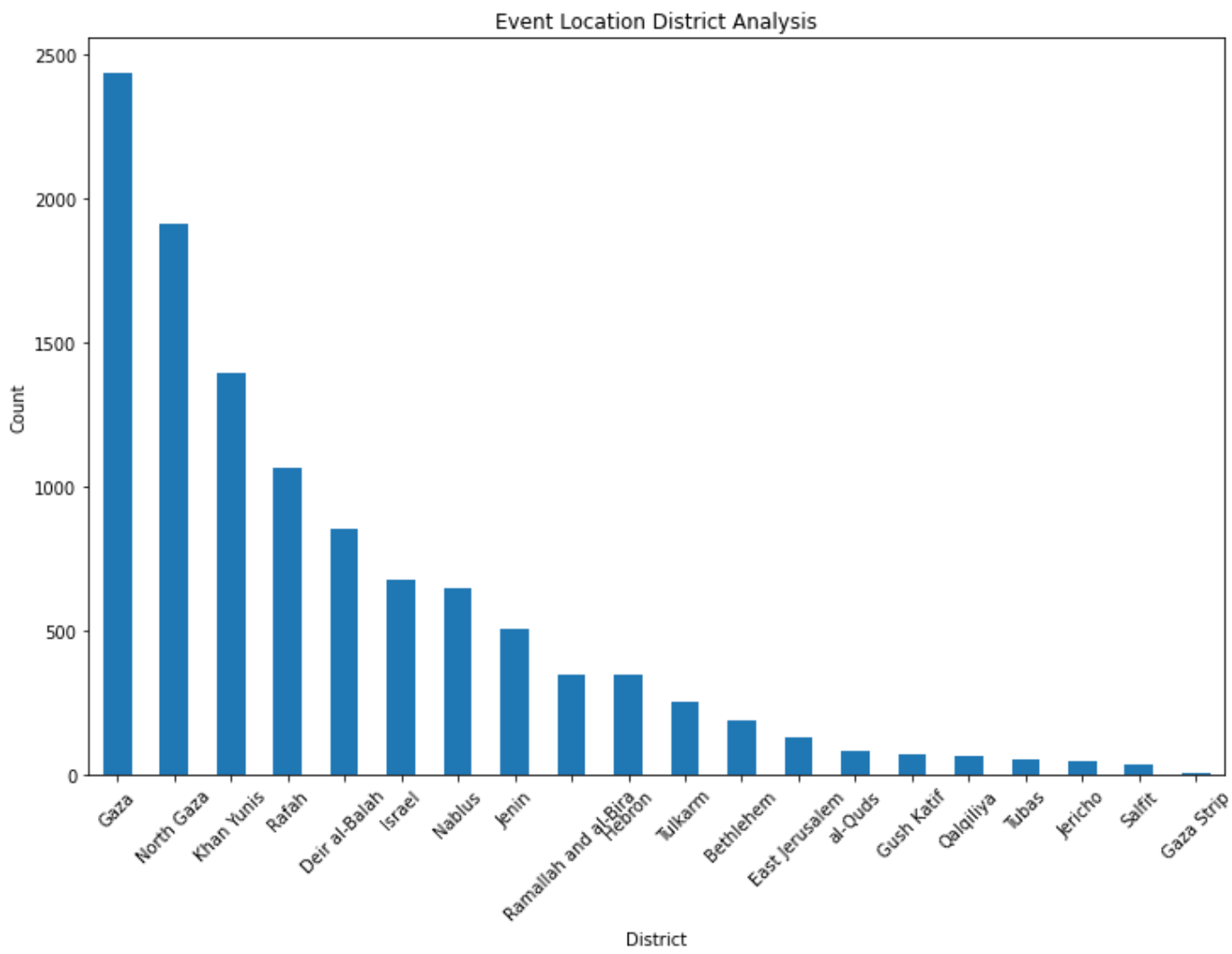


In [44]:
```
def event_location_analysis(df):
    fig = px.scatter_geo(df,
```

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

event_location_analysis(df)
```

In [45]:

```
def event_location_district_analysis(df):
    plt.figure(figsize=(12, 8))
    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_district_analysis(df)
```

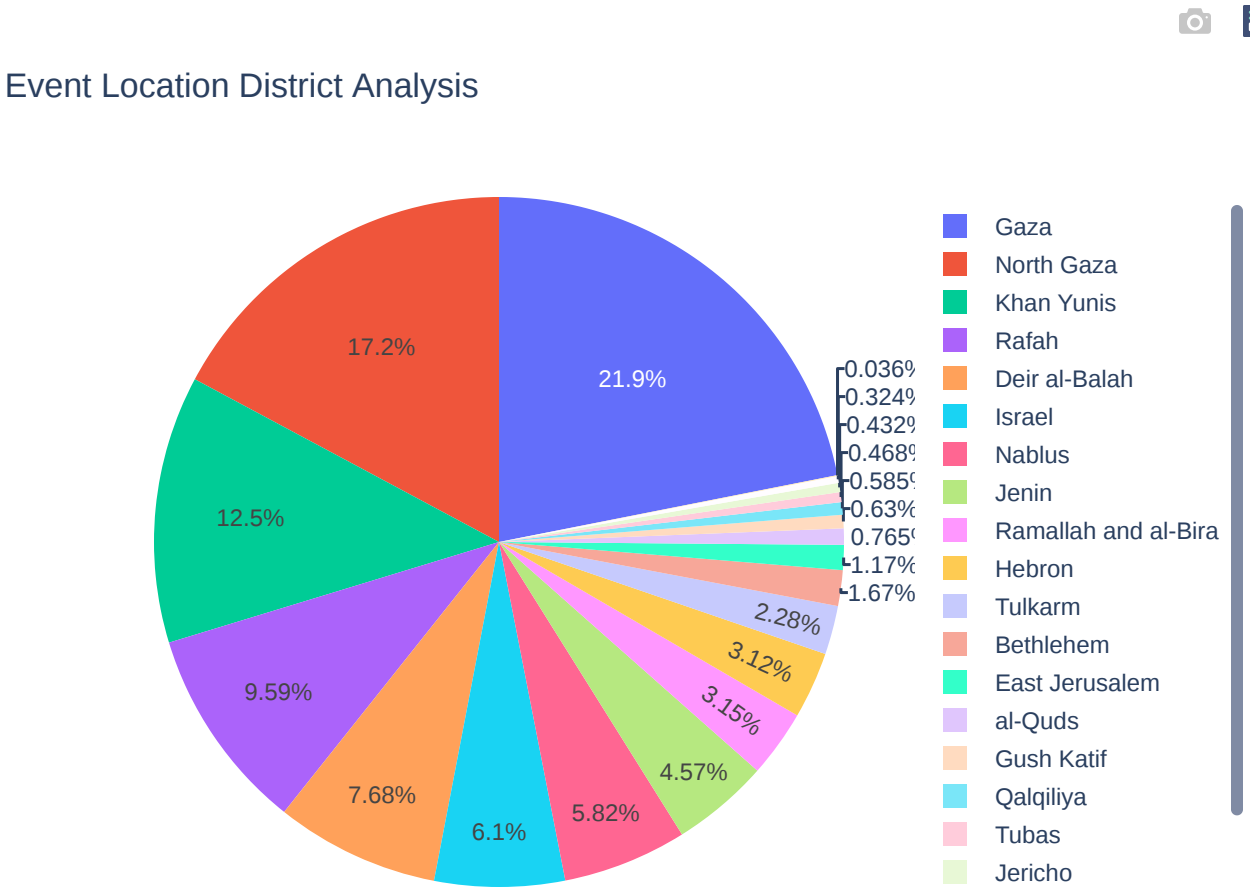


In [46]:

```python
def event_location_district_analysis(df):
    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)
```

In [47]:

```python
def event_location_district_analysis(df):

    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)
```



In [48]:

```python
def name_analysis(df):

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

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

    return missing_names_count, invalid_names
```

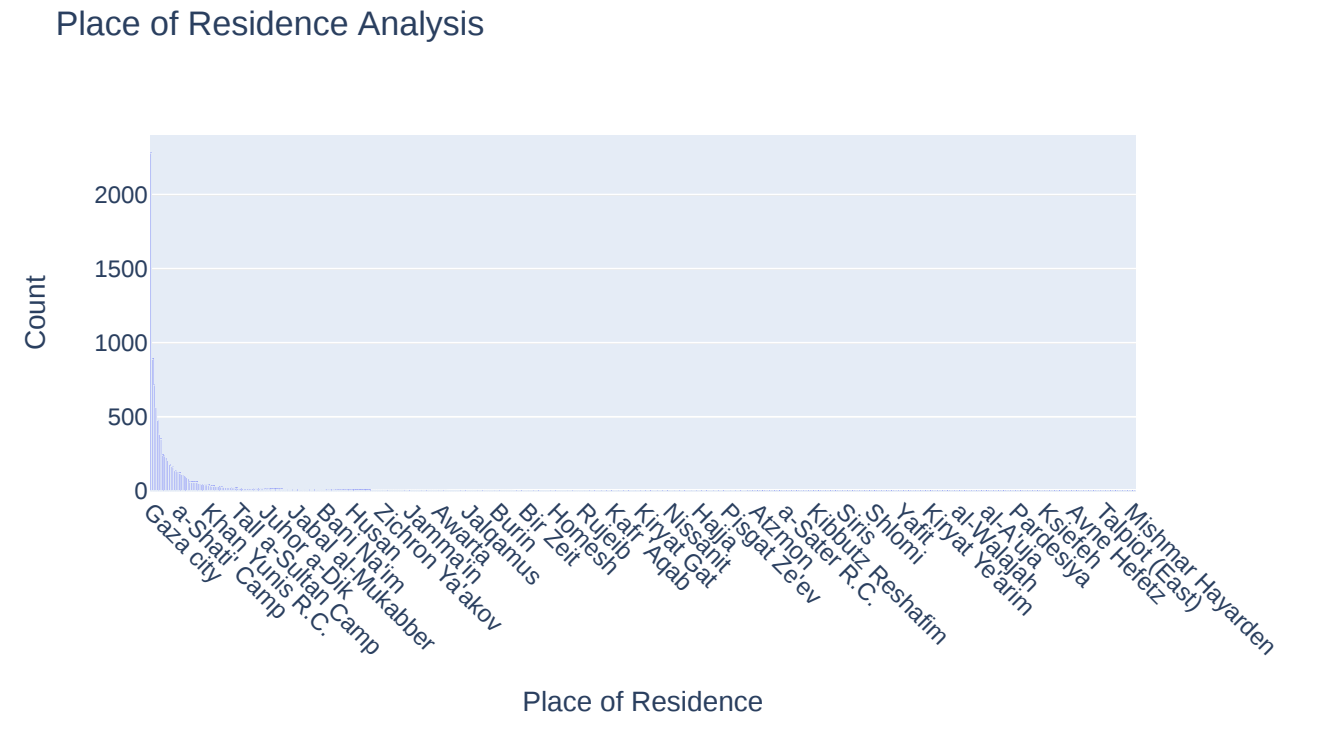
```
missing_names_count, invalid_names = name_analysis(df)

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

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

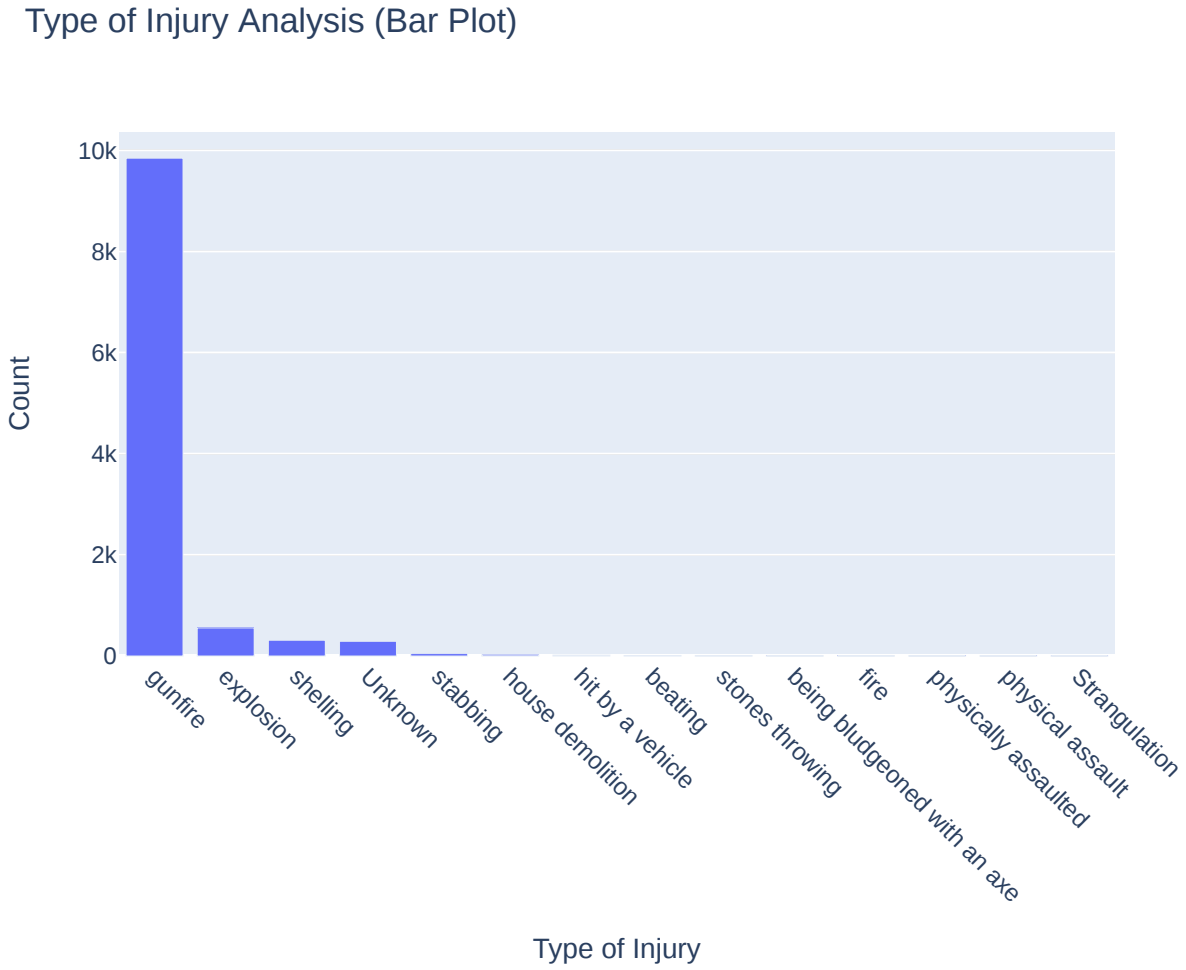
In [49]:
```
print("Invalid Names:")
invalid_names
```

Out[49]:
```
Invalid Names:
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
```

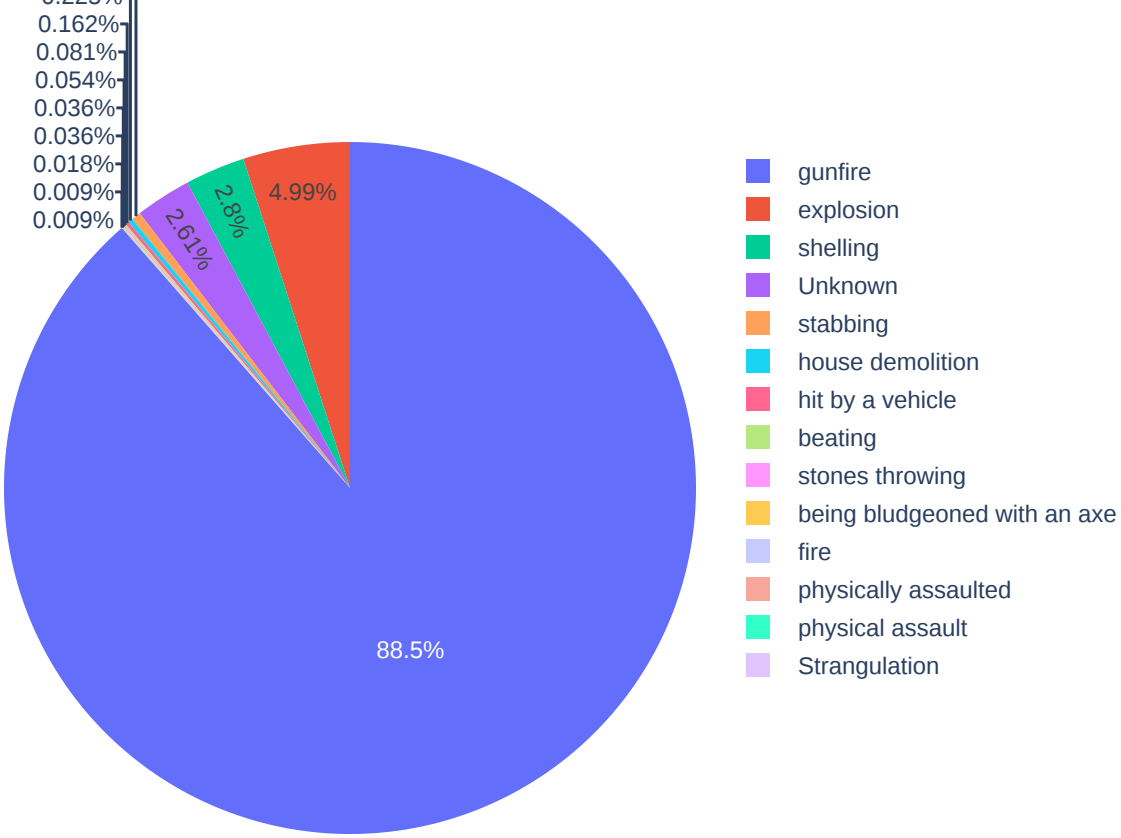
In [50]:
```
def place_of_residence_analysis(df):

    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_residence_analysis(df)
```



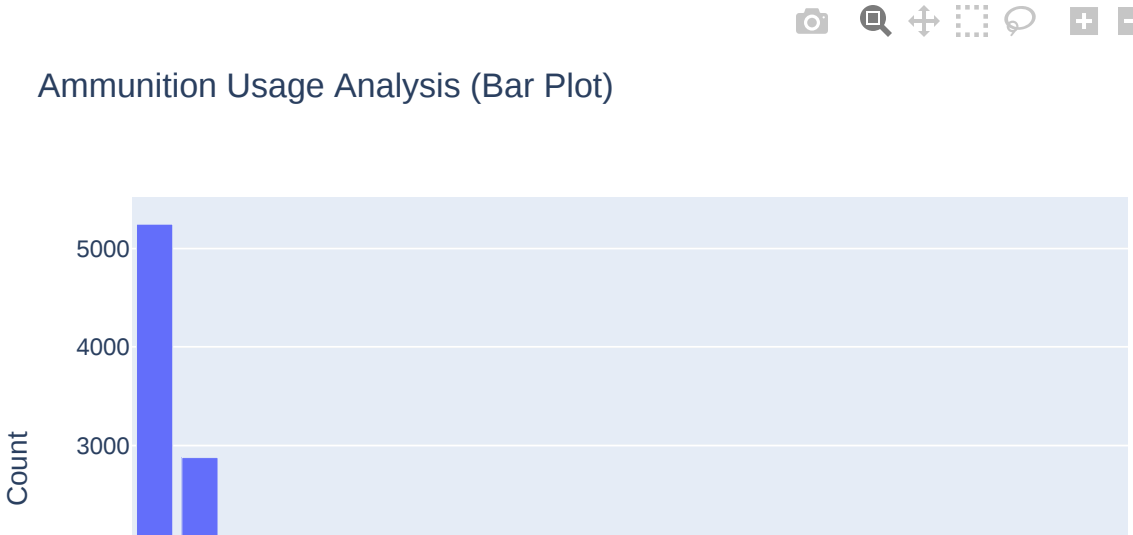
In [51]:
```python
def type_of_injury_analysis_bar(df):

    injury_counts = df['type_of_injury'].value_counts()

    fig = px.bar(injury_counts,
                 x=injury_counts.index,
                 y=injury_counts.values,
                 title='Type of Injury Analysis (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)
```

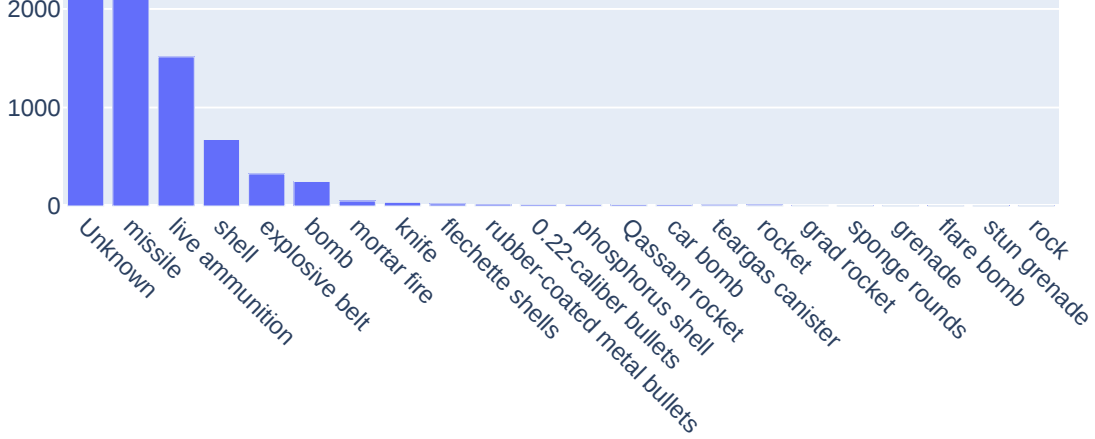


In [52]:
```python
def type_of_injury_analysis(df):

    injury_counts = df['type_of_injury'].value_counts()

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

    fig.show()

type_of_injury_analysis(df)
```

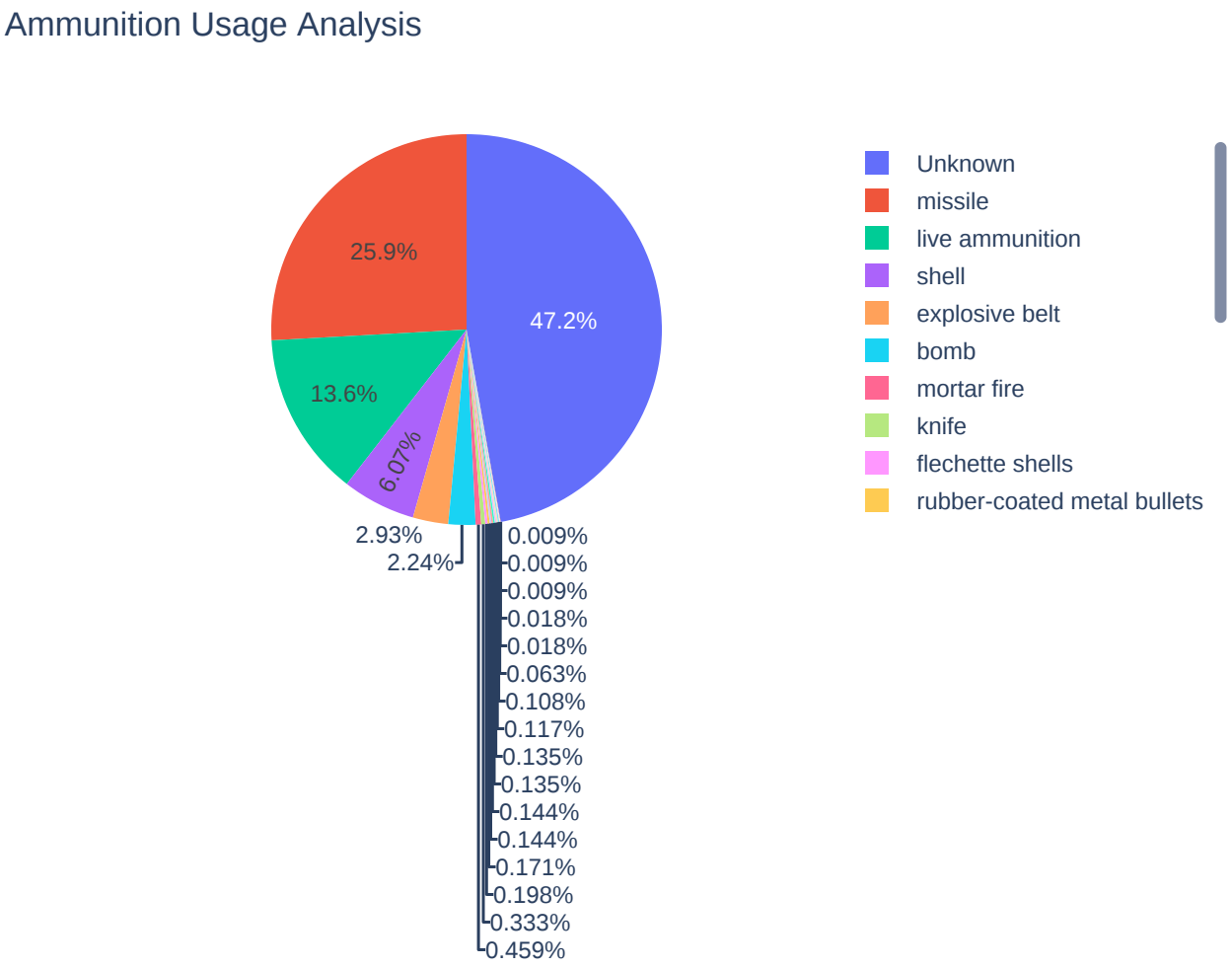


In [53]:
```python
def ammunition_analysis_bar(df):

    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': 'Ammunition Type', 'y': 'Count'})

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

    fig.show()

ammunition_analysis_bar(df)
```





In [54]:
```
def ammunition_analysis(df):

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

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

    fig.show()

ammunition_analysis(df)
```

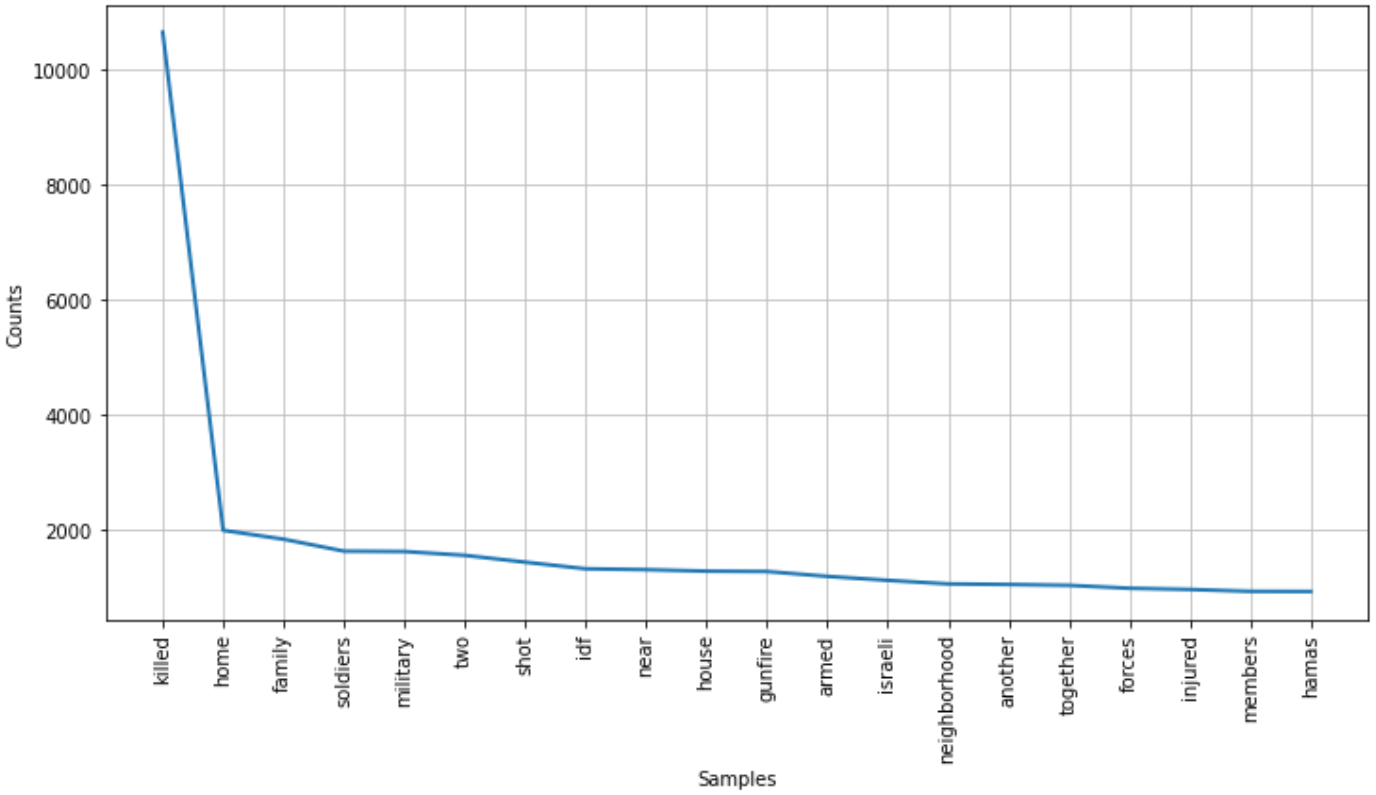


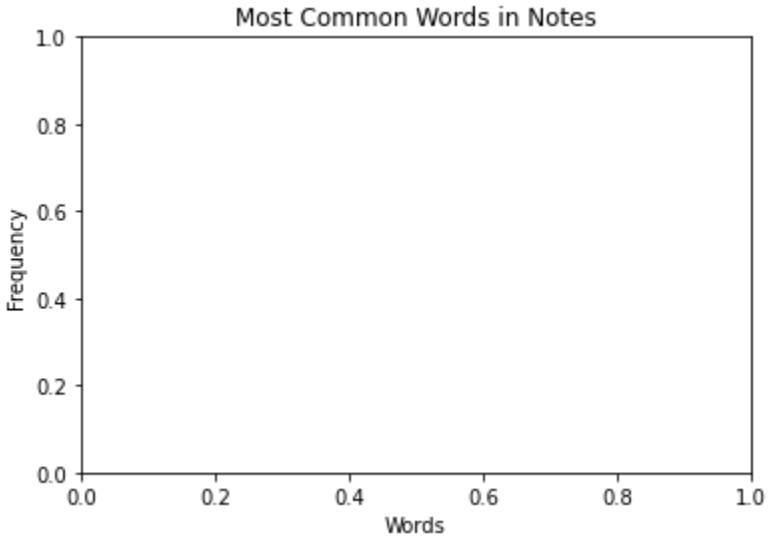
In [55]:
```
import nltk
```

```
from nltk import FreqDist
from nltk.tokenize import word_tokenize
```

In [56]:
```
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()

    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()
```

In [57]:
```
notes_word_frequency_analysis(df)
```





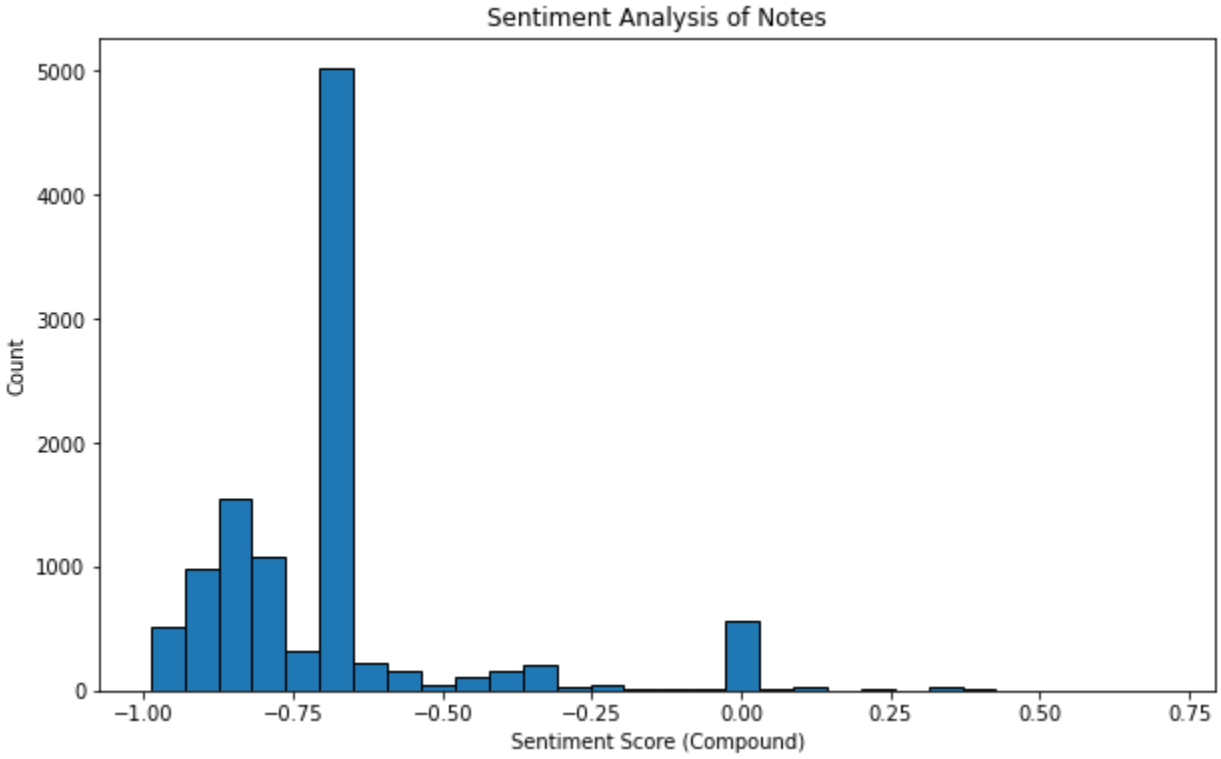
In [58]:
```
def notes_word_frequency_analysis(df):

    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()

    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', 1
612), ('two', 1545), ('shot', 1426), ('idf', 1311), ('near', 1296), ('house', 1270), ('g
unfire', 1264), ('armed', 1179), ('israeli', 1110), ('neighborhood', 1048), ('another',
1038), ('together', 1023), ('forces', 972), ('injured', 951), ('members', 918), ('hama
s', 916)]
```

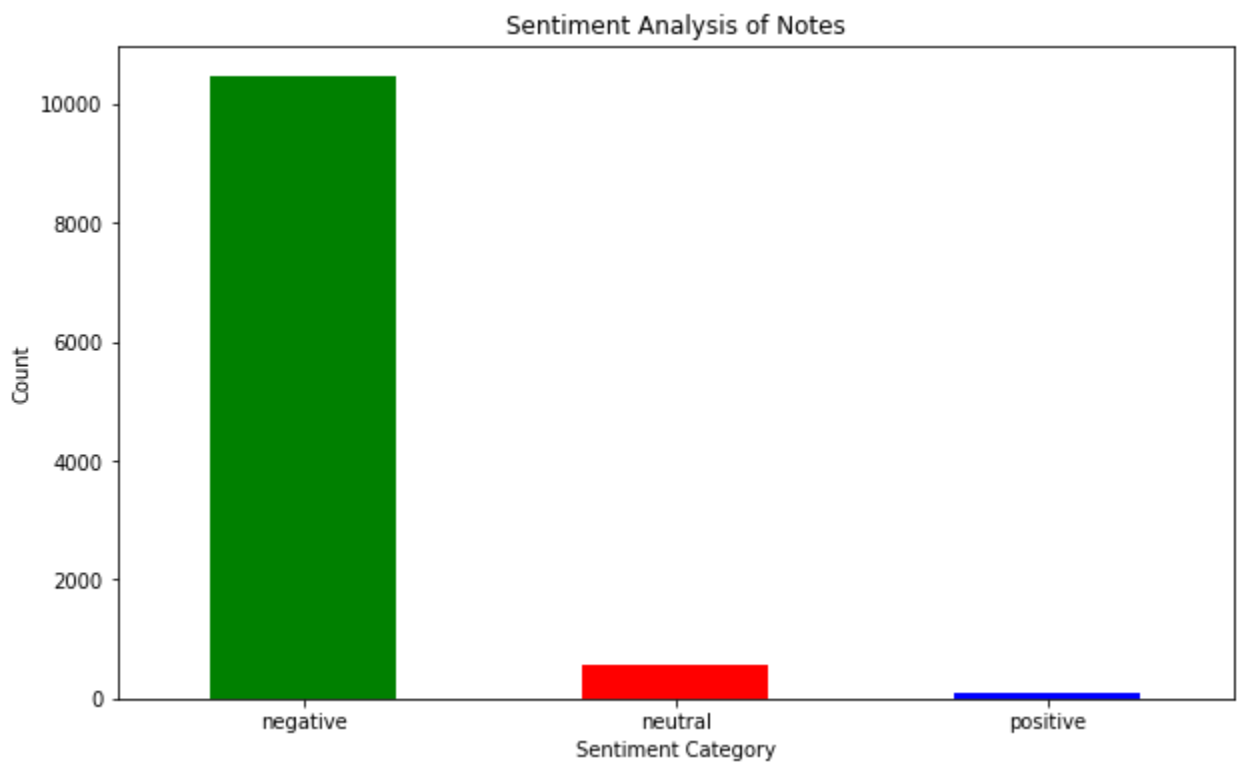
In [59]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

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

    sid = SentimentIntensityAnalyzer()

    df['sentiment_score'] = df['notes'].apply(lambda x: sid.polarity_scores(str(x))['com

    plt.figure(figsize=(10, 6))
    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()
```

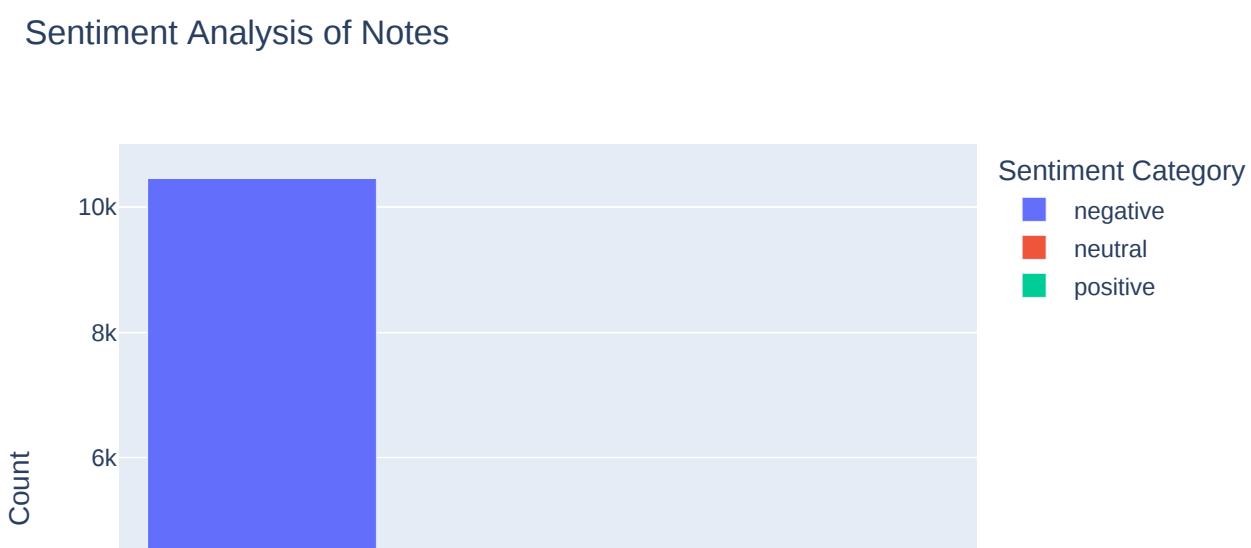
In [61]:
```
sentiment_analysis(df)
```

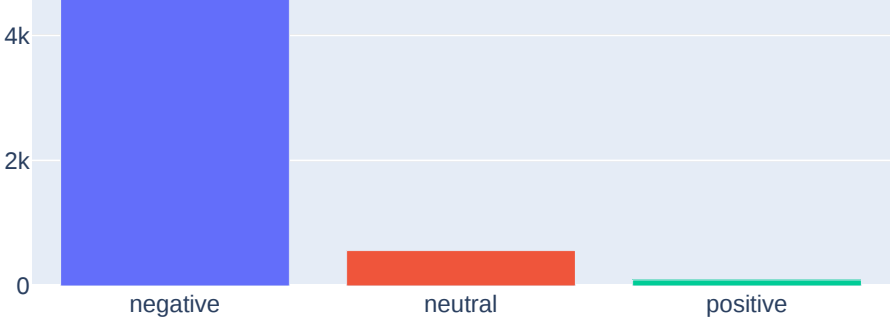


In [62]:
```
def sentiment_analysis(df):

    sid = SentimentIntensityAnalyzer()

    df['sentiment_score'] = df['notes'].apply(lambda x: sid.polarity_scores(str(x))['com

    df['sentiment_category'] = df['sentiment_score'].apply(lambda score: 'positive' if s

    plt.figure(figsize=(10, 6))
    df['sentiment_category'].value_counts().plot(kind='bar', color=['green', 'red', 'blu
    plt.title('Sentiment Analysis of Notes')
    plt.xlabel('Sentiment Category')
    plt.ylabel('Count')
    plt.xticks(rotation=0)
    plt.show()

sentiment_analysis(df)
```

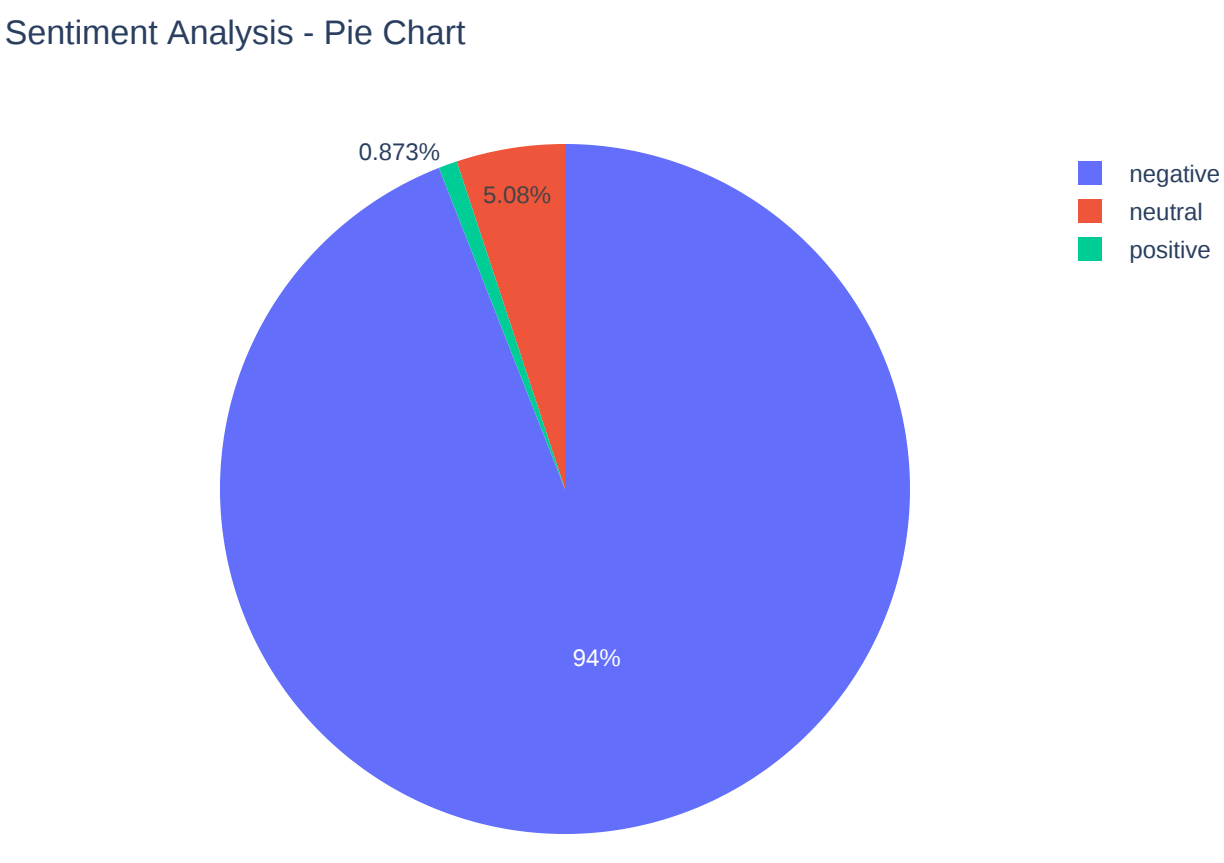


In [63]:
```
def sentiment_analysis_plotly(df):

    sid = SentimentIntensityAnalyzer()

    df['sentiment_score'] = df['notes'].apply(lambda x: sid.polarity_scores(str(x))['com

    df['sentiment_category'] = df['sentiment_score'].apply(lambda score: 'positive' if s

    sentiment_counts = df['sentiment_category'].value_counts().reset_index()
    sentiment_counts.columns = ['Sentiment Category', 'Count']

    fig = px.bar(sentiment_counts,
                 x='Sentiment Category',
                 y='Count',
                 color='Sentiment Category',
                 title='Sentiment Analysis of Notes',
                 labels={'Sentiment Category': 'Sentiment Category', 'Count': 'Count'})

    fig.show()

sentiment_analysis_plotly(df)
```



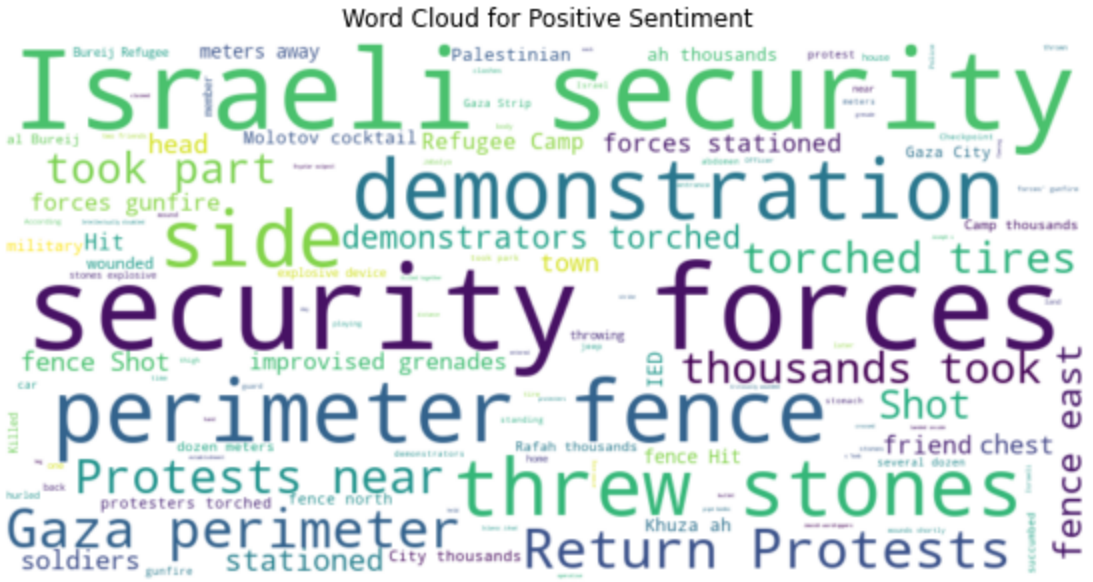


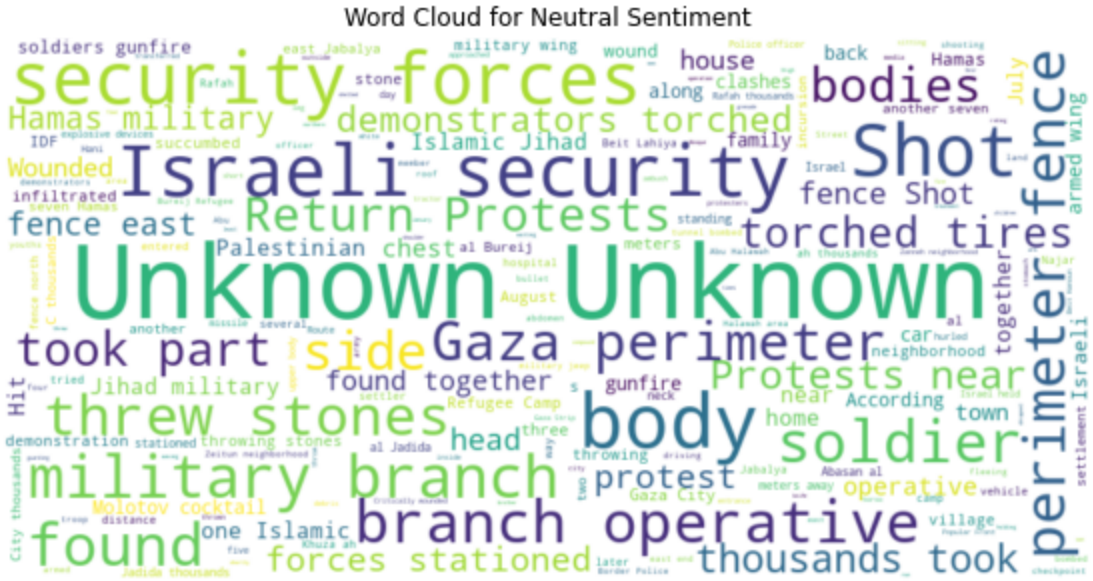
In [64]:
```
def sentiment_analysis_pie_plotly(df):

    sid = SentimentIntensityAnalyzer()

    df['sentiment_score'] = df['notes'].apply(lambda x: sid.polarity_scores(str(x))['com

    df['sentiment_category'] = df['sentiment_score'].apply(lambda score: 'positive' if s

    sentiment_distribution = df['sentiment_category'].value_counts()

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

    fig.show()

sentiment_analysis_pie_plotly(df)
```



In [65]:

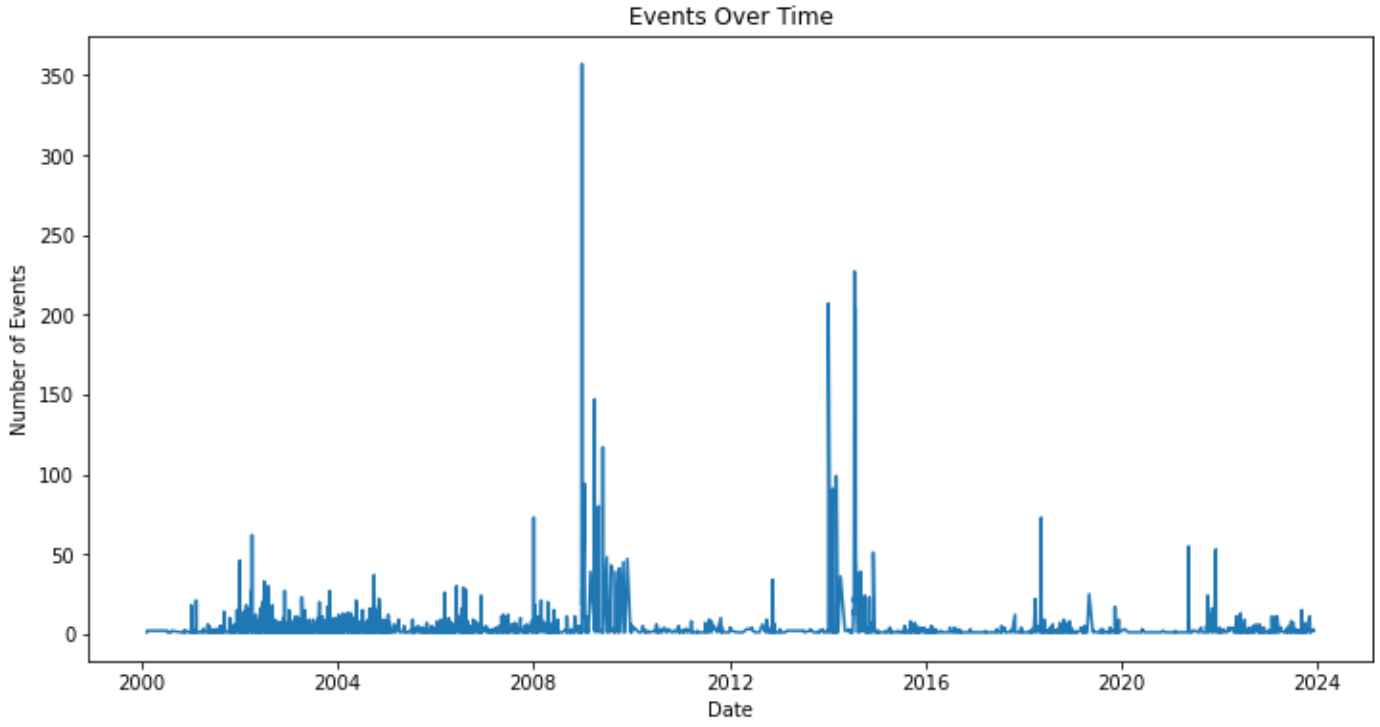
```
def generate_wordcloud(sentiment_category, notes):

    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)

generate_wordclouds_for_sentiments(df)
```





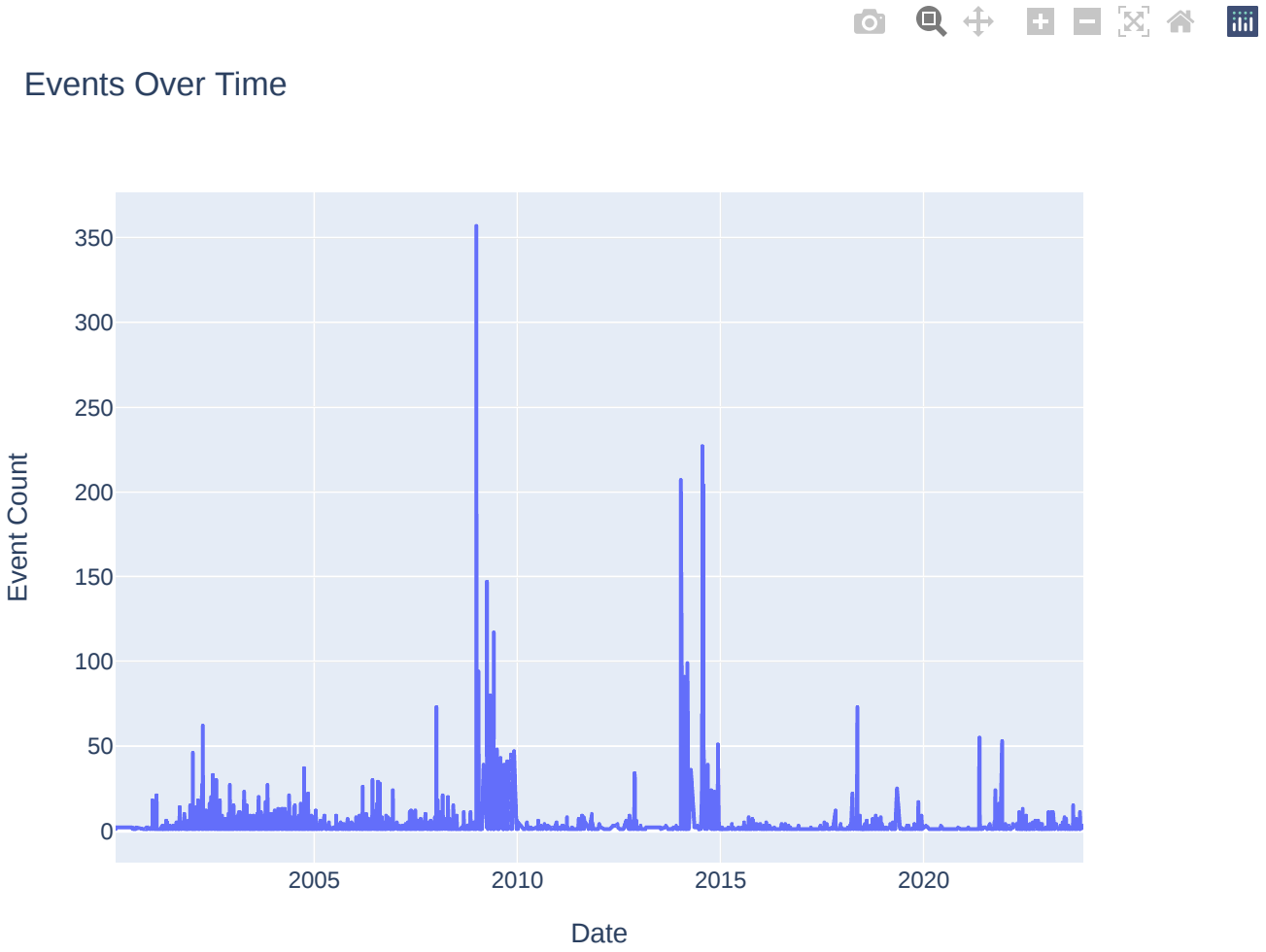
In [66]:
```
df['date_of_event'] = pd.to_datetime(df['date_of_event'])
```

In [67]:
```
def time_based_analysis(df):

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

    plt.figure(figsize=(12, 6))
    events_by_date.plot()
    plt.title('Events Over Time')
    plt.xlabel('Date')
    plt.ylabel('Number of Events')
    plt.show()

time_based_analysis(df)
```



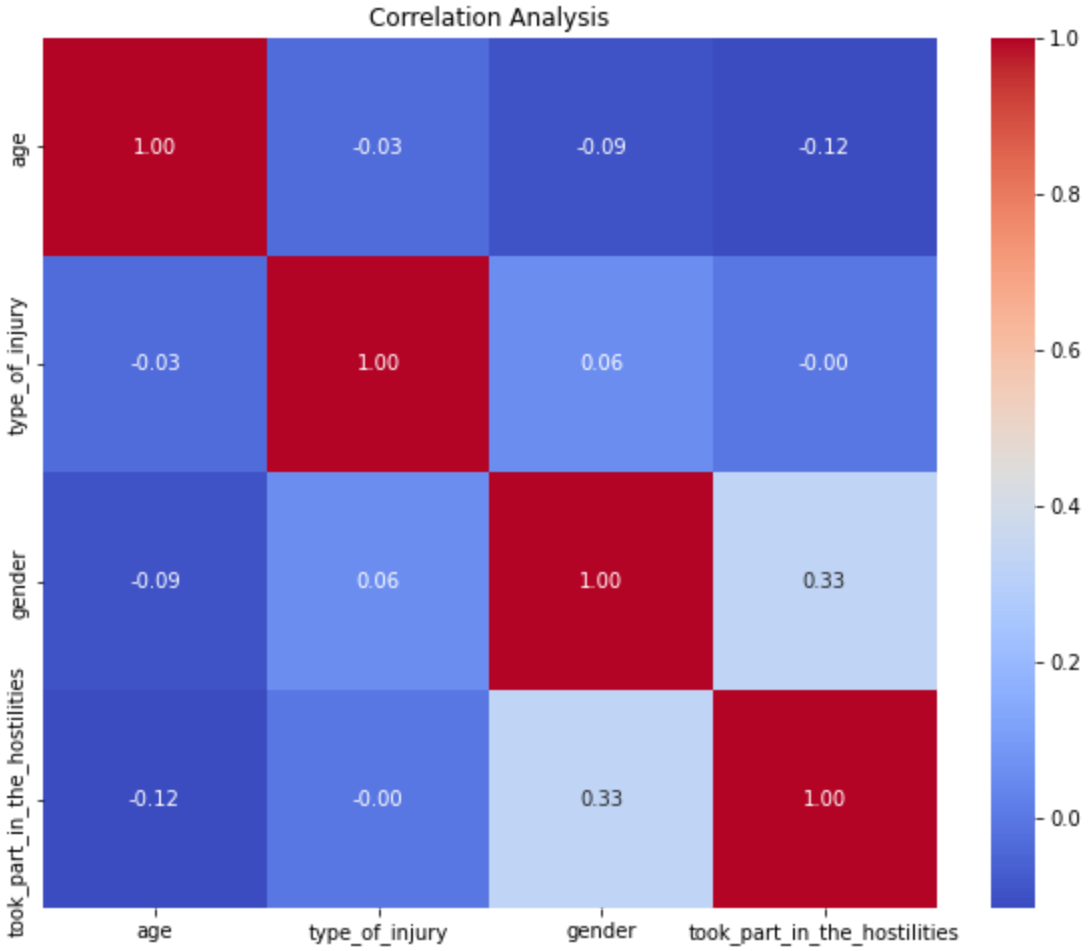
In [68]:
```
def time_based_analysis_plotly(df):

    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='Events Over Time')
```

```
    fig.show()

time_based_analysis_plotly(df)
```



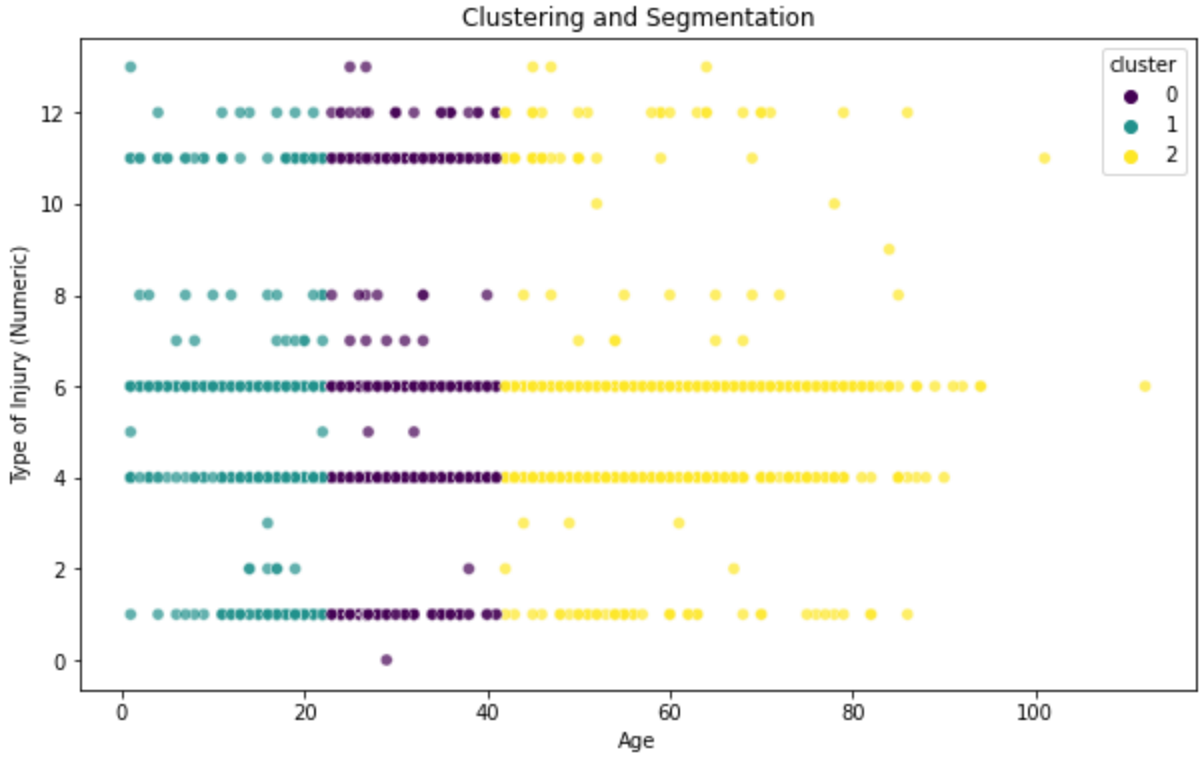
In [69]:

```
def correlation_analysis(df):

    selected_columns = ['age', 'type_of_injury', 'gender', 'took_part_in_the_hostilities
    selected_df = df[selected_columns]

    selected_df['gender'] = selected_df['gender'].astype('category').cat.codes
    selected_df['type_of_injury'] = selected_df['type_of_injury'].astype('category').cat
    selected_df['took_part_in_the_hostilities'] = selected_df['took_part_in_the_hostilit

    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()

correlation_analysis(df)
```



In [70]:
```
from sklearn.cluster import KMeans
```

In [71]:
```
def clustering_and_segmentation(df):

    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('catego

    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=(10, 6))
    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)
```



Thanks !!!

In [72]:
```
# Project by: Prof. Nirmal Gaud
# Contact: ds.ml.projects.sessions.1@gmail.com
# WhatssApp Group (Join for ML/DL Projects): https://chat.whatsapp.com/BQ0vLtxjVS3I1M9Yd
```