```
df = df.drop(columns=["Unnamed: 68"])
```

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

	Country	Code	Indicator_name	Indicator_code	1960	1961	1962	1963	1964	1965	...	2014	2015	2016	2017	
0	Aruba	ABW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1	Africa Eastern and Southern	AFE	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	3559470.0	4195068.0	4917140.0	6058091.0	595
2	Afghanistan	AFG	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	2596259.0	2666294.0	2501447.0	2624265.0	268
3	Africa Western and Central	AFW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	919751.0	1038377.0	1090726.0	1155918.0	128

1) This line removes (deletes) the column named "Unnamed: 68" from the DataFrame df and saves the updated DataFrame back into df

```
import pandas as pd #Used for data handling and analysis.
import matplotlib.pyplot as plt #Used for creating graphs and plots.
import numpy as np
import geopandas as gpd
import seaborn as sns #Used for advanced and attractive data visualizations.
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv("Refugee_Dataset.csv", skiprows=4)
```

```
df
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2015	2016	2017	2018	
0	Aruba	ABW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1	Africa Eastern and	AFE	Refugee population by country	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	4195068.0	4917140.0	6058091.0	5958200.0	593

1.1)Removing demographic groupings

```
df.drop(df[df['Country'].str.contains('demographic dividend')].index, inplace=True)
```

```
df
```

	Country	Code	Indicator_name	Indicator_code	1960	1961	1962	1963	1964	1965	...	2014	2015	2016	2017
0	Aruba	ABW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1	Africa Eastern and Southern	AFE	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	3559470.0	4195068.0	4917140.0	6058091.0	5
2	Afghanistan	AFG	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	2596259.0	2666294.0	2501447.0	2624265.0	2
3	Africa Western and Central	AFW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	919751.0	1038377.0	1090726.0	1155918.0	1

1.2) Filling missing values

Creates a list of strings representing years from "1960" to "1989"

```
df = df.drop(columns=[str(year) for year in range(1960, 1990)])
```

```
Fills missing values in columns from the 5th onward with the mean of each column.
```

```
df.iloc[:, 4:] = df.iloc[:, 4:].T.fillna(df.iloc[:, 4:].T.mean()).T
df
```

	Country	Code	Indicator_name	Indicator_code	1990	1991	1992	1993	1994	1995	...	2014	201
0	Aruba	ABW	Refugee population by country or territory of ...	SM.POP.REFG.OR	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2	Afghanistan	AFG	Refugee population by country or territory of ...	SM.POP.REFG.OR	6.339095e+06	6.306301e+06	4.552153e+06	3.374575e+06	2.731166e+06	2.679132e+06	...	2596259.0	2666294.(
4	Angola	AGO	Refugee population by country or territory of ...	SM.POP.REFG.OR	4.077600e+05	3.816380e+05	3.004940e+05	3.238220e+05	2.825700e+05	2.466530e+05	...	9468.0	11855.(
5	Albania	ALB	Refugee population by country or territory of ...	SM.POP.REFG.OR	1.822000e+03	3.542000e+03	4.355000e+03	4.747000e+03	5.021000e+03	5.804000e+03	...	10156.0	10404.(

1.3)Keeps only the rows where both 1999 and 2019 columns have data (removes rows with missing values in these columns).

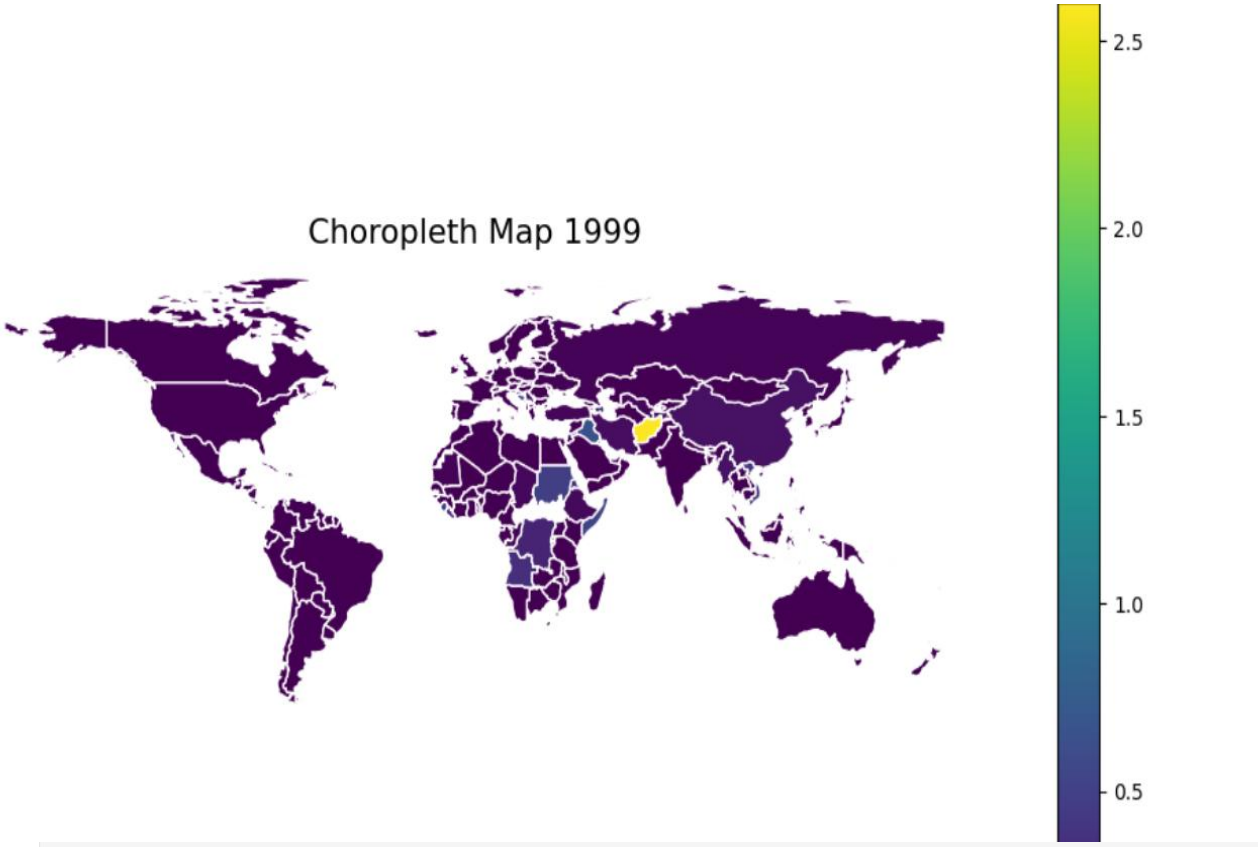
```
]: df = df[df['1999'].notna() & df['2019'].notna()]
   df
```

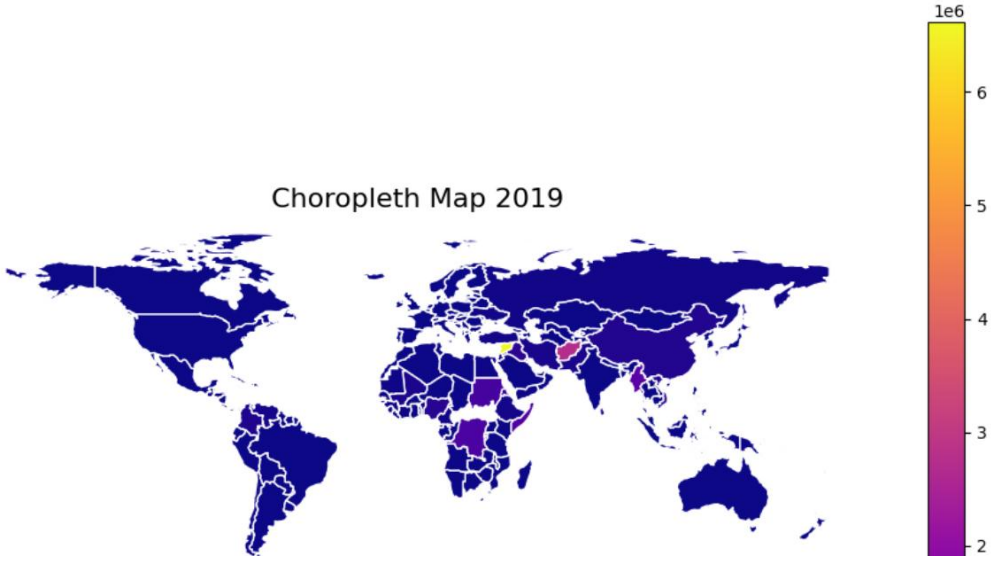
]:

	Country	Code	Indicator_name	Indicator_code	1990	1991	1992	1993	1994	1995	...	2014	
2	Afghanistan	AFG	Refugee population by country or territory of ...	SM.POP.REFG.OR	6.339095e+06	6.306301e+06	4.552153e+06	3.374575e+06	2.731166e+06	2.679132e+06	...	2596259.0	2666
4	Angola	AGO	Refugee population by country or territory of ...	SM.POP.REFG.OR	4.077600e+05	3.816380e+05	3.004940e+05	3.238220e+05	2.825700e+05	2.466530e+05	...	9468.0	11
5	Albania	ALB	Refugee population by country or	SM.POP.REFG.OR	1.822000e+03	3.542000e+03	4.355000e+03	4.747000e+03	5.021000e+03	5.804000e+03	...	10156.0	10

```python
# Convert to GeoDataFrame
merged_df = gpd.GeoDataFrame(merged_df)

# Create figure and axes
fig, ax = plt.subplots(figsize=(12, 8))

# Plot choropleth map for 1999 with a new color map
merged_df.plot(
    column="1999",              # Column to visualize
    ax=ax,
    legend=True,                # Show legend
    cmap="viridis",             # Attractive color map
    missing_kwds={"color": "lightgrey"},  # Color for missing data
    edgecolor="white"           # Country borders color
)

# Add title
ax.set_title("Choropleth Map 1999", fontsize=16)

# Remove axes for cleaner look
ax.set_axis_off()

# Show plot
plt.show()
```

The choropleth map illustrates that countries experiencing sustained high levels of conflict since 1999 also record large refugee populations in 2019. This spatial pattern highlights the cumulative effect of prolonged violence on forced displacement, where long-term instability generates persistent refugee flows rather than temporary migration. The results reinforce the strong association between chronic conflict exposure and large-scale humanitarian displacement.


Choropleth Map 1999
2.5
2.0
1.5
1.0
0.5


Choropleth Map 2019
1e6
6
5
4
3
2

TASK 2.2.2

Middle East countries Total refugee population > 50,000 (2019) Conflict intensity choropleth

Step 1. Define Middle East countries

Natural Earth does not provide a clean Middle East flag, so we define it explicitly.

```
# Convert latitude and longitude to numeric, invalid values become NaN
df[['latitude', 'longitude']] = df[['latitude', 'longitude']].apply(pd.to_numeric, errors='coerce')
```

```
# Convert death columns to numeric
death_columns = ['deaths_a', 'deaths_b', 'deaths_civilians', 'deaths_unknown', 'best', 'high', 'low']
for col in death_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)
```

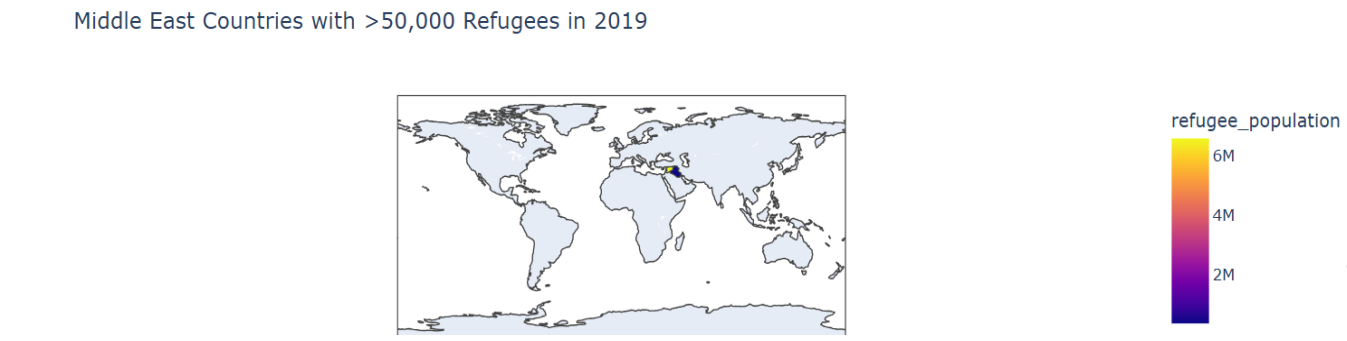
```
# Create total deaths column
df['total_deaths'] = df['best']
```

```
# Filter relevant years (1990-2023)
df = df[(df['year'] >= 1990) & (df['year'] <= 2023)]
df
```

	id	relid	year	active_year	code_status	type_of_violence	conflict_dset_id	conflict_new_id	conflict_name	dyad_dset_id	...	deaths_a	deaths_b	deaths_
0	244657	IRQ-2017-1-524-322	2017	1	Clear	1	259	259	Iraq: Government	524	...	0	4	
1	412700	IRQ-2021-1-	2021	1	Clear	1	259	259	Iraq:	524	...	13	1	

```
import plotly.express as px

# Reset the index for a clean DataFrame
ref2019_clean = ref2019.reset_index(drop=True)

# Create an interactive choropleth map for refugee population in 2019
fig = px.choropleth(
    ref2019_clean,
    locations='Code',                    # Country codes for mapping
    color='refugee_population',          # Column to determine color intensity
    hover_name='Country',                # Country name shown on hover
    color_continuous_scale='turbo',      # Attractive and vibrant color scale
    title='Refugee Population (2019) - Countries with >5000 Conflicts Since 1999'
)

# Display the interactive map
fig.show()
```

```python
import plotly.express as px

# Create an interactive choropleth map for Middle East countries with >50,000 refugees
fig = px.choropleth(
    me_ref2019,
    locations='Code',                    # Country codes for mapping
    color='refugee_population',          # Column to determine color intensity
    hover_name='Country',                # Country name shown on hover
    color_continuous_scale='plasma',     # Attractive and vibrant color scale
    title='Middle East Countries with >50,000 Refugees in 2019'
)

# Display the interactive map
fig.show()
```

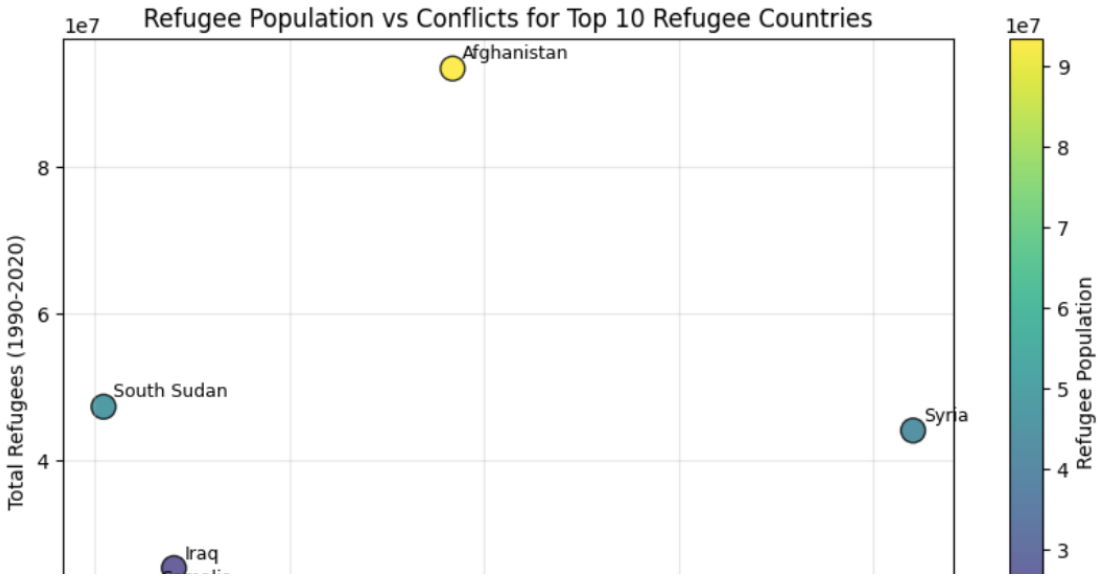



```python
# Define a mapping to standardize country names
mapping = {
    'Congo, Dem. Rep.': 'DR Congo (Zaire)',
    'Myanmar': 'Myanmar (Burma)',
    'Viet Nam': 'Vietnam'
}

# Apply the mapping to the top 10 refugee countries
top_10_mapped = [mapping.get(country, country) for country in top_10['Country'].tolist()]

# Filter conflict data for the top 10 countries between 1990 and 2020
con = conflict[
    (conflict['country'].isin(top_10_mapped)) &
    (conflict['year'] >= 1990) &
    (conflict['year'] <= 2020)
]

# Calculate total conflicts per country and sort in descending order
total_conflicts = con.groupby('country')['conflict_count'].sum().sort_values(ascending=False)

# Display the results
print("Total conflicts for top 10 refugee countries (1990-2020):")
print(total_conflicts)
```

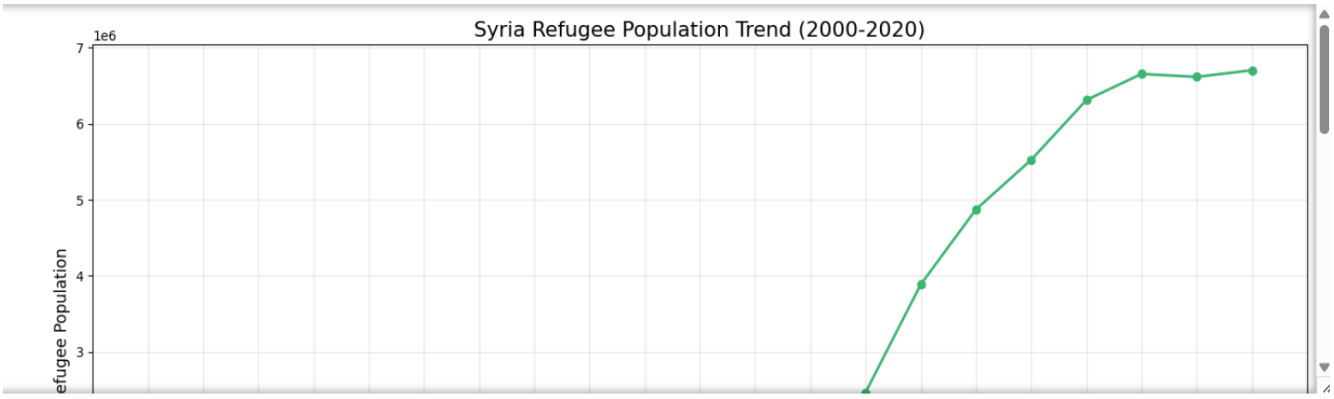
```
Total conflicts for top 10 refugee countries (1990-2020):
country
Syria                84132
Afghanistan          36781
Iraq                  8121
```

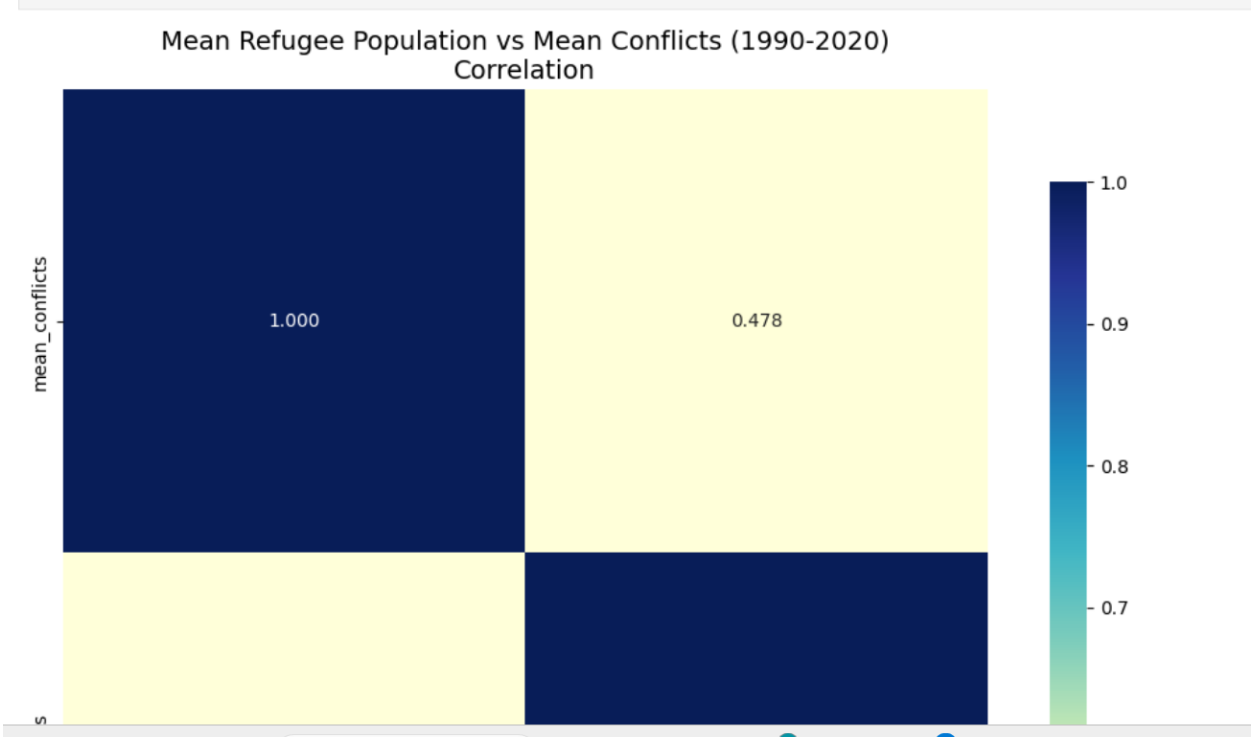
```python
import matplotlib.pyplot as plt

# Align data - only countries that appear in both datasets
common_countries = set(top_10['Country']) & set(total_conflicts.index)
refugee_values = [top_10[top_10['Country'] == country]['refugees_1990_2020'].iloc[0] for country in common_countries]
conflict_values = [total_conflicts[country] for country in common_countries]

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

# Scatter plot with color and alpha
plt.scatter(
    conflict_values,
    refugee_values,
    s=150,                # Bubble size
    c=refugee_values,     # Color based on refugee population
    cmap='viridis',       # Attractive color map
    alpha=0.8,            # Transparency
    edgecolor='black'     # Border color for better contrast
)

# Add country labels
for i, country in enumerate(common_countries):
    plt.annotate(
        country,
        (conflict_values[i], refugee_values[i]),
        xytext=(5, 5),
        textcoords='offset points',
        fontsize=9
    )
```

```python
plt.xlabel('Total Conflicts (1990-2020)')
plt.ylabel('Total Refugees (1990-2020)')
plt.title('Refugee Population vs Conflicts for Top 10 Refugee Countries')
plt.grid(True, alpha=0.3)
plt.colorbar(label='Refugee Population')  # Show color scale
plt.show()
```

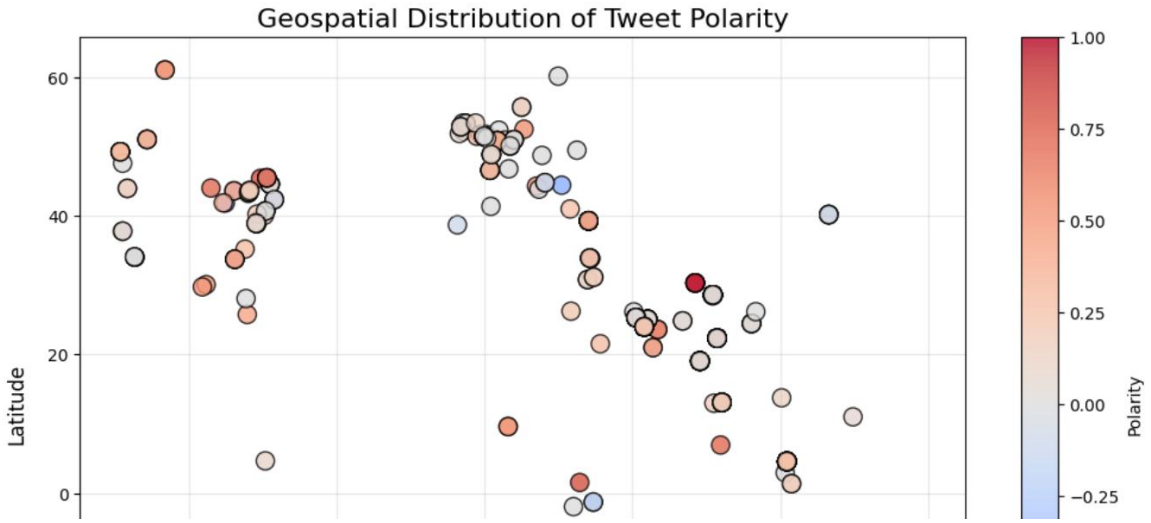


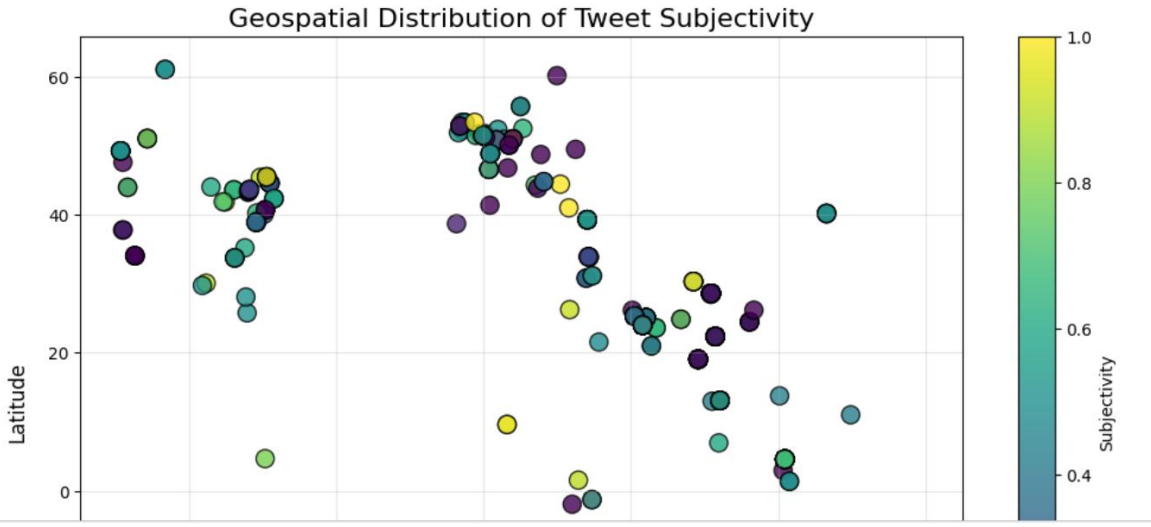
```
plt.title('Syria Refugee Population Trend (2000-2020)', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Refugee Population', fontsize=12)
plt.grid(True, alpha=0.3)
plt.xticks(trend_df['year'], rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping
plt.show()

# Display the data
print("Syria refugee trend (2000-2020):")
print(trend_df)
```





```
# Display polarity statistics
print("Polarity statistics:")
print(geocoded_df['polarity'].describe())
```





192]:

	Unnamed: 0	tweets	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date
0	0	ChatGPT: Optimizing Language Models for Dialog...	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	08/04/2009 17:52	405	1692	3247	False	20/12/2020 06:06
1	1	Try talking with ChatGPT, our new AI system wh...	Albert Fong	San Francisco, CA	Marketing dude, tech geek, heavy metal & '80s ...	21/09/2009 15:27	834	666	178	False	13/12/2020 16:27
2	2	ChatGPT: Optimizing Language Models for Dialog...	elıʇeu 👌	Your Bed	heil, hydra 🖐 ☺	25/06/2020 23:30	10	88	155	False	12/12/2020 20:33
3	3	THRILLED to share that ChatGPT, our new	Charles Adler	Vancouver, BC - Canada	Hosting "CharlesAdlerTonight" Global News Radi...	10/09/2008 11:28	49165	3933	21853	True	12/12/2020 20:23