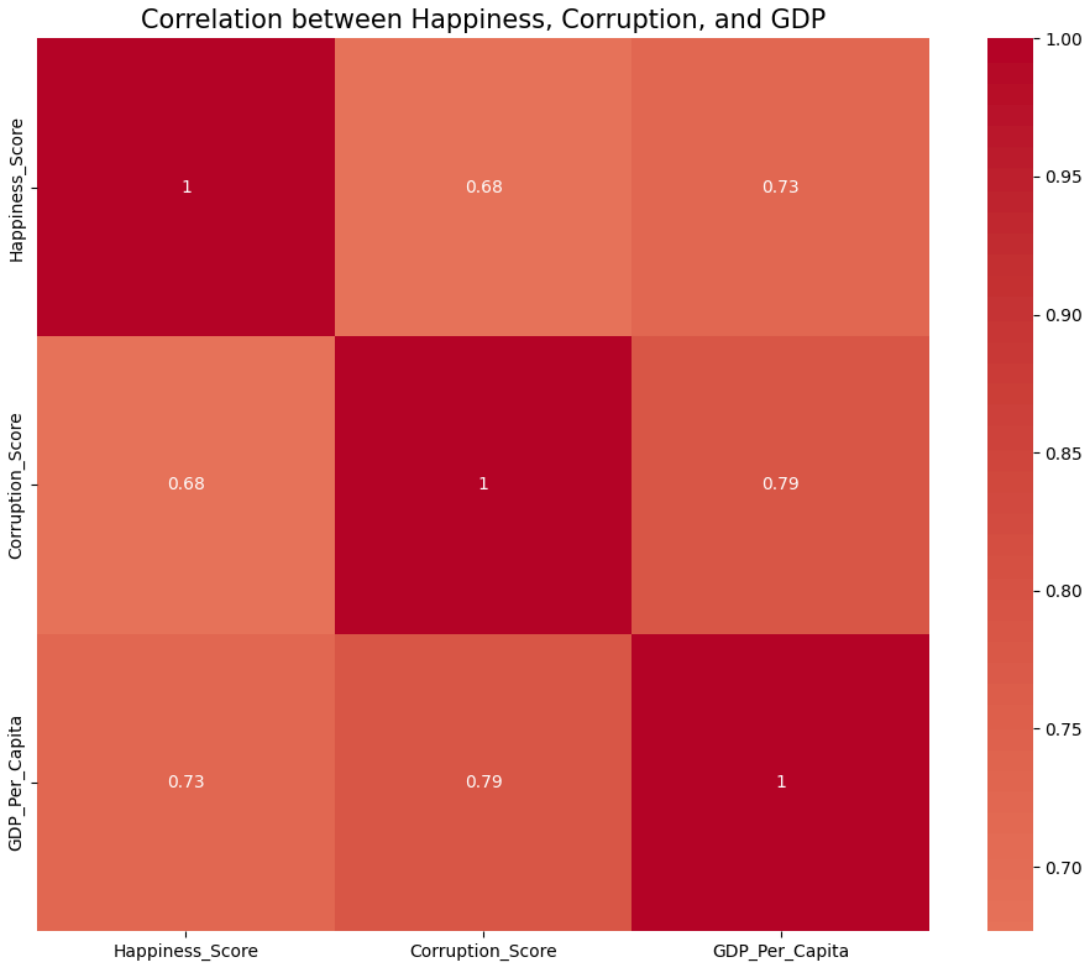
Assignment 5
Mihael Zlatev - 1MI3400543

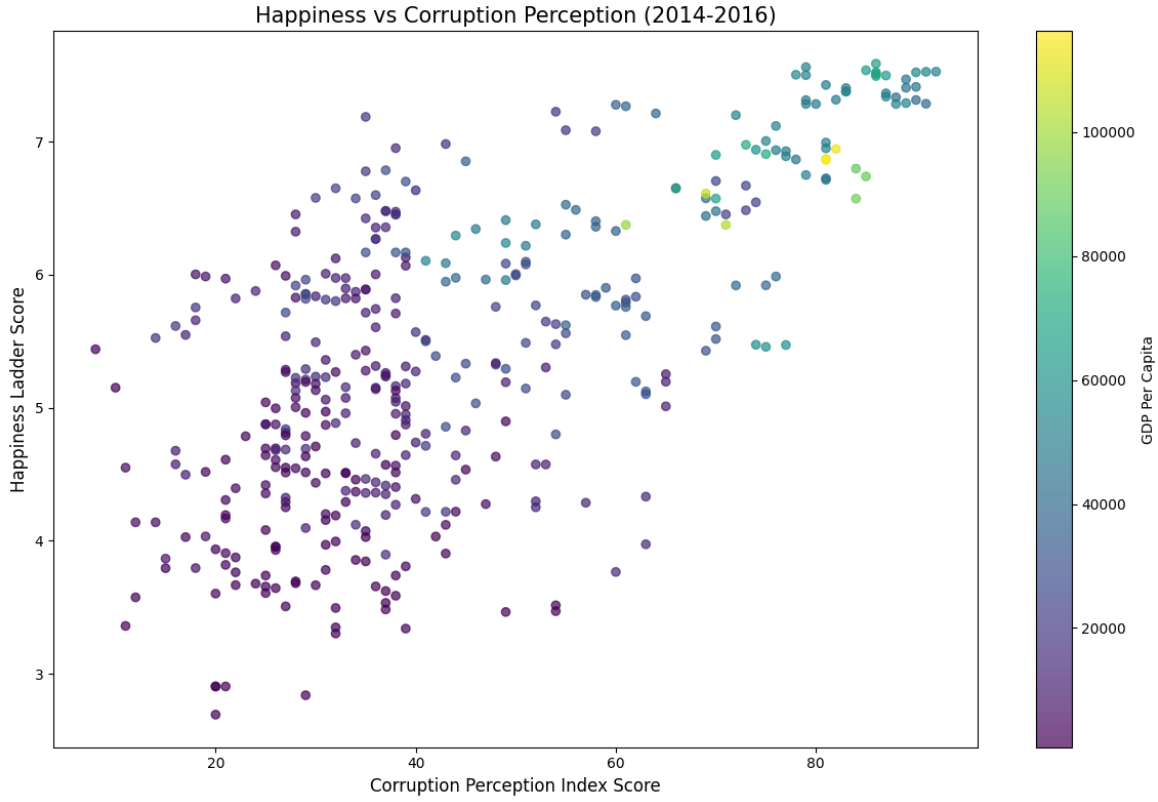
Task 1 - Analysis of Happiness, Corruption and GDP

1. Heatmap that visualizes the correlations between Happiness Score, Corruption Score, and GDP per Capita.



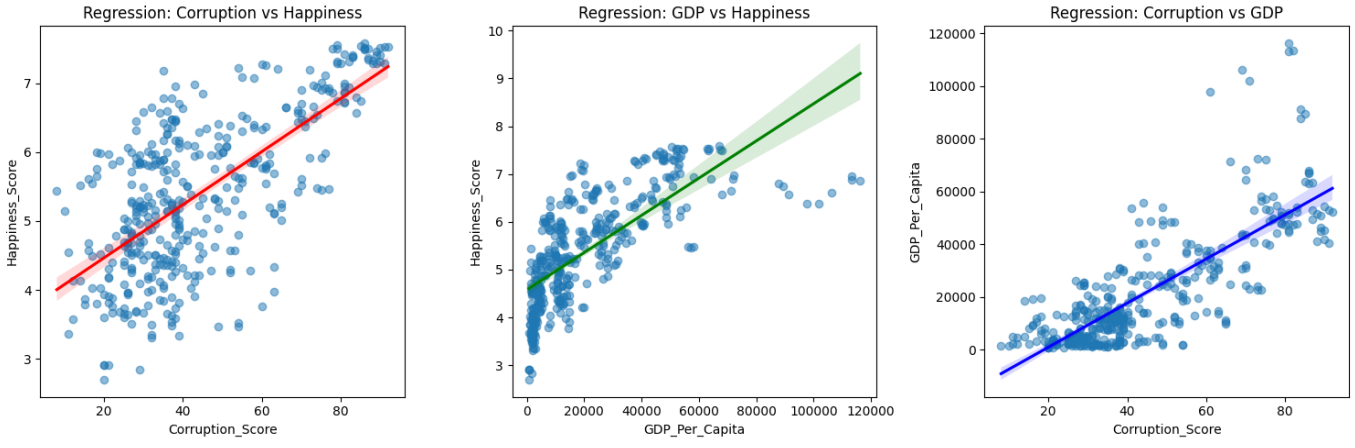
The heatmap reveals strong correlations between the three variables: Happiness Score, Corruption Score, and GDP per Capita. The data suggests that higher levels of happiness are associated with lower levels of corruption and higher GDP per capita, while higher corruption is linked to lower GDP per capita.

2. Happiness vs Corruption Perception



The scatter plot includes data only 2014, 2015 and 2016 because the dataset for the corruption and happiness fully intercept on on those years.
The data points are widely scattered, indicating a strong negative correlation between happiness score and corruption perception. Countries with higher corruption perception (higher index scores) tend to have lower happiness scores, and vice versa.
It can be also seen that higher GDP indicates a lower levels of curruption.
However, there are some outliers which shows desptite a relatively high level of curription people still can feel happy.

3. Regression plots



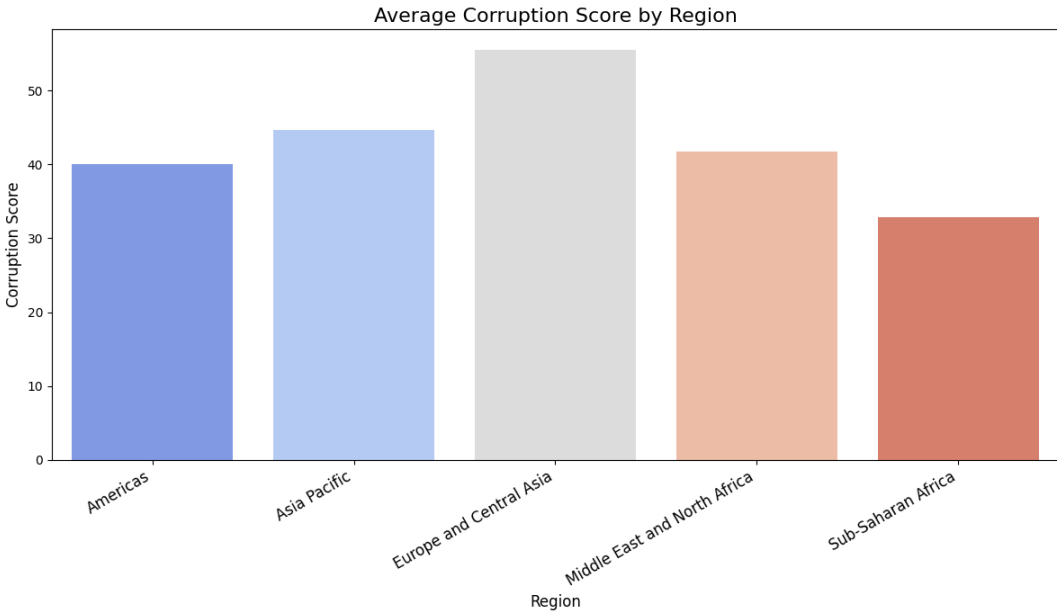
- Corruption vs Happiness: The regression line shows a strong negative linear relationship,

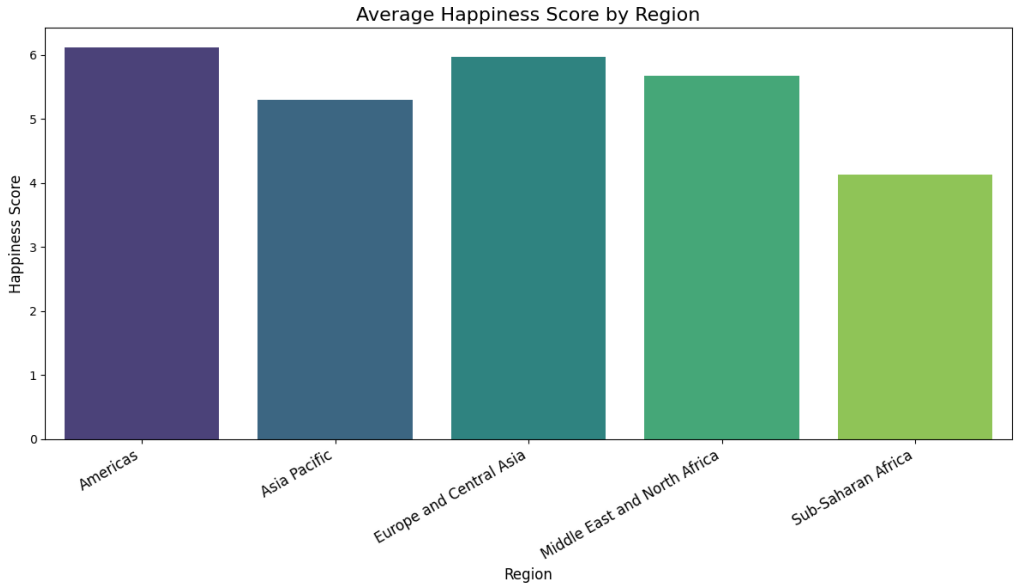
confirming the findings from the previous scatterplot. As corruption decreases, happiness tends to increase.

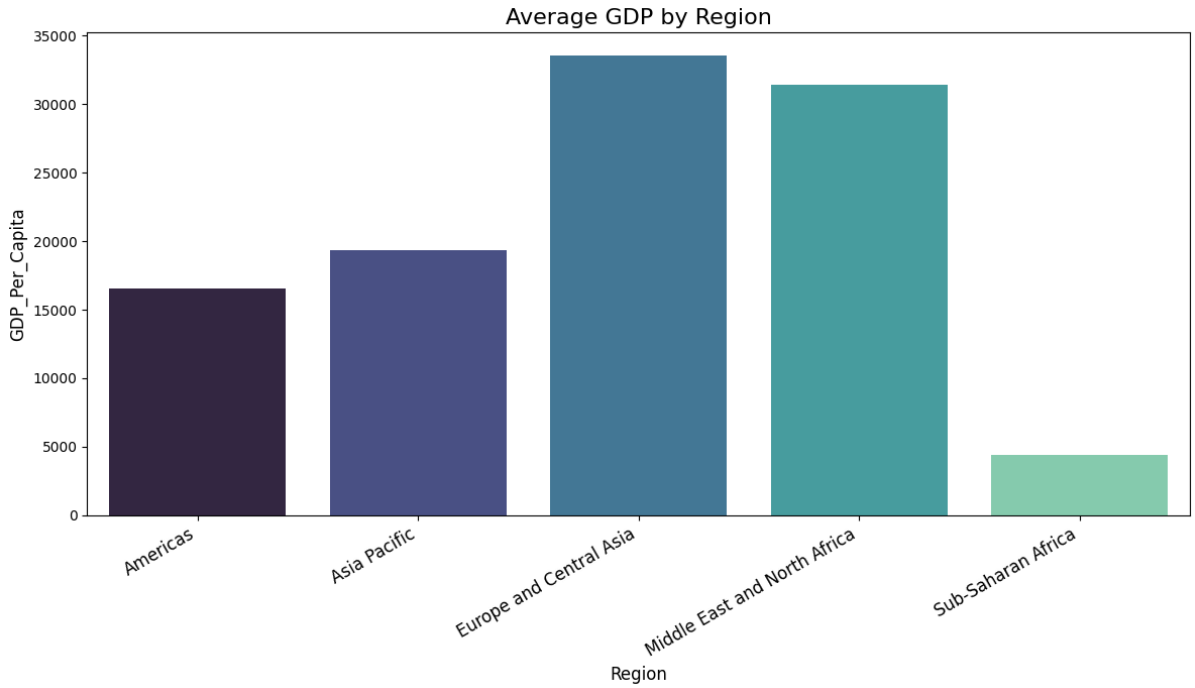
- GDP per Capita vs Happiness: The regression line shows a strong positive linear relationship. Higher GDP per capita is associated with higher happiness scores.

- Corruption vs GDP per Capita: The regression line shows a strong negative linear relationship. Higher levels of corruption are linked to lower GDP per capita.

4. Average corruption, happiness, GDP per Region







From those average we can easily conclude that the highest *corruption (low index indicates high corruption)* is in Africa while having the lowest levels of happiness and GDP. This again confirms the analysis of the previous plots. However, overall, no matter in which region the average people feel relatively happy. Even in Sub-Saharan Africa the happiness index is ~4.5 while the highest index is in Americas at around 6.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns

TARGET_REGIONS = [
    "Americas",
    "Asia Pacific",
    "Europe and Central Asia",
    "Middle East and North Africa",
    "Sub-Saharan Africa"
]

def init_data(corruption_df, gdp_happiness_df):
    corruption_long = pd.melt(
        corruption_df,
        id_vars=['Country', 'Country Code', 'Region'],
        value_vars=['CPI 2014 Score', 'CPI 2015 Score', 'CPI 2016 Score'],
        var_name='Year',
        value_name='Corruption_Score'
    )
    corruption_long['Year'] =
corruption_long['Year'].str.extract(r'(\d+)').astype(int)
```

```
    corruption_long = corruption_long.rename(columns={
        'Country': 'Entity',
        'Country Code': 'Code'
    })

    # Preprocess GDP and Happiness Data
    gdp_happiness_filtered = gdp_happiness_df[
        (gdp_happiness_df['Year'].isin([2014, 2015, 2016])) &
        (gdp_happiness_df['Cantril ladder score'].notna()) &
        (gdp_happiness_df['GDP per capita, PPP (constant 2017 international
$)'].notna())
    ]
    gdp_happiness_filtered = gdp_happiness_filtered.rename(columns={
        'Cantril ladder score': 'Happiness_Score',
        'GDP per capita, PPP (constant 2017 international $)': 'GDP_Per_Capita'
    })

    # Merge datasets
    return pd.merge(
        gdp_happiness_filtered,
        corruption_long,
        on=['Entity', 'Code', 'Year'],
        how='inner'
    )

def analyze_data(df):
    plot_heatmap(df)
    scatter_plot(df)
    plot_regressions(df)

def plot_heatmap(df):
    plt.figure(figsize=(10, 8))
    correlation_matrix = df[['Happiness_Score', 'Corruption_Score',
'GDP_Per_Capita']].corr()
    sns.heatmap(
        correlation_matrix,
        annot=True,
        cmap='coolwarm',
        center=0,
        square=True
    )
    plt.title('Correlation between Happiness, Corruption, and GDP', fontsize=15)
    plt.tight_layout()
    plt.show()

def scatter_plot(df):
    plt.figure(figsize=(12, 8))
    scatter = plt.scatter(
        df['Corruption_Score'],
        df['Happiness_Score'],
        c=df['GDP_Per_Capita'],
```

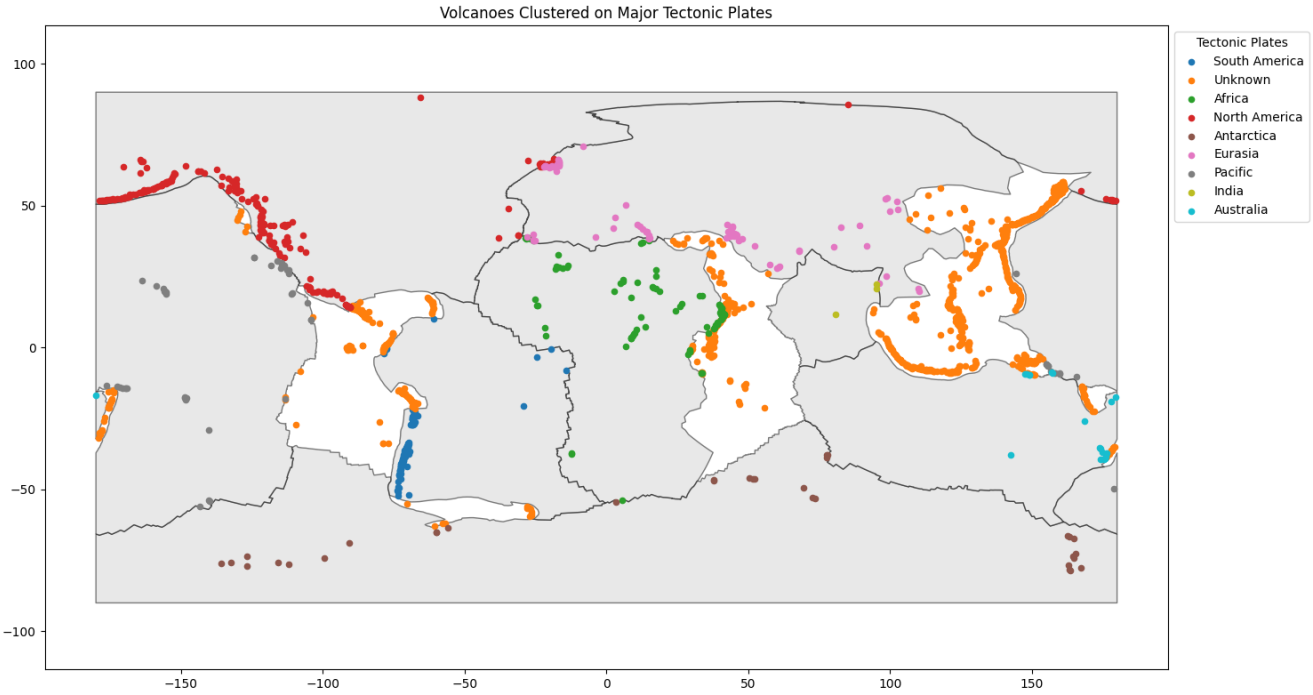
```
        cmap='viridis',
        alpha=0.7
    )
    plt.colorbar(scatter, label='GDP Per Capita')
    plt.title('Happiness vs Corruption Perception (2014-2016)', fontsize=15)
    plt.xlabel('Corruption Perception Index Score', fontsize=12)
    plt.ylabel('Happiness Ladder Score', fontsize=12)
    plt.tight_layout()
    plt.show()

def plot_regressions(df):
    plt.figure(figsize=(15, 5))
    plt.subplot(1, 3, 1)
    sns.regplot(
        x='Corruption_Score',
        y='Happiness_Score',
        data=df,
        scatter_kws={'alpha':0.5},
        line_kws={'color': 'red'}
    )
    plt.title('Regression: Corruption vs Happiness')

    plt.subplot(1, 3, 2)
    sns.regplot(
        x='GDP_Per_Capita',
        y='Happiness_Score',
        data=df,
        scatter_kws={'alpha':0.5},
        line_kws={'color': 'green'}
    )
    plt.title('Regression: GDP vs Happiness')

    plt.subplot(1, 3, 3)
    sns.regplot(
        x='Corruption_Score',
        y='GDP_Per_Capita',
        data=df,
        scatter_kws={'alpha':0.5},
        line_kws={'color': 'blue'}
    )
    plt.title('Regression: Corruption vs GDP')
    plt.tight_layout()
    plt.show()


def analyze_data_per_regions(df):
    regional_data = df[df['Region'].isin(TARGET_REGIONS)]

    region_analysis = regional_data.groupby('Region').agg({
        'Happiness_Score': 'mean',
        'Corruption_Score': 'mean',
```

```
        'GDP_Per_Capita': 'mean'
    }).reset_index()

    plot_bars(
        data=region_analysis,
        x='Region',
        y='Happiness_Score',
        palette='viridis',
        title='Average Happiness Score by Region',
        ylabel='Happiness Score'
    )

    plot_bars(
        data=region_analysis,
        x='Region',
        y='Corruption_Score',
        palette='coolwarm',
        title='Average Corruption Score by Region',
        ylabel='Corruption Score'
    )

    plot_bars(
        data=region_analysis,
        x='Region',
        y='GDP_Per_Capita',
        palette='mako',
        title='Average GDP by Region',
        ylabel='GDP_Per_Capita'
    )


def plot_bars(data, x, y, palette, title, ylabel):
    plt.figure(figsize=(12, 7))
    sns.barplot(
        data=data,
        x=x,
        y=y,
        palette=palette,
        hue=x,
        legend=False
    )
    plt.title(title, fontsize=16)
    plt.xticks(rotation=30, fontsize=12, ha='right')
    plt.ylabel(ylabel, fontsize=12)
    plt.xlabel(x, fontsize=12)
    plt.tight_layout()
    plt.show()


if __name__=="__main__":
    corruption_df = pd.read_csv("corruption.csv")
```

```
gdp_vs_happiness_df = pd.read_csv("gdp-vs-happiness.csv")
data = init_data(corruption_df, gdp_vs_happiness_df)
analyze_data(data)
analyze_data_per_regions(data)
```

Task 2 – Clustering of volcanoes on Earth

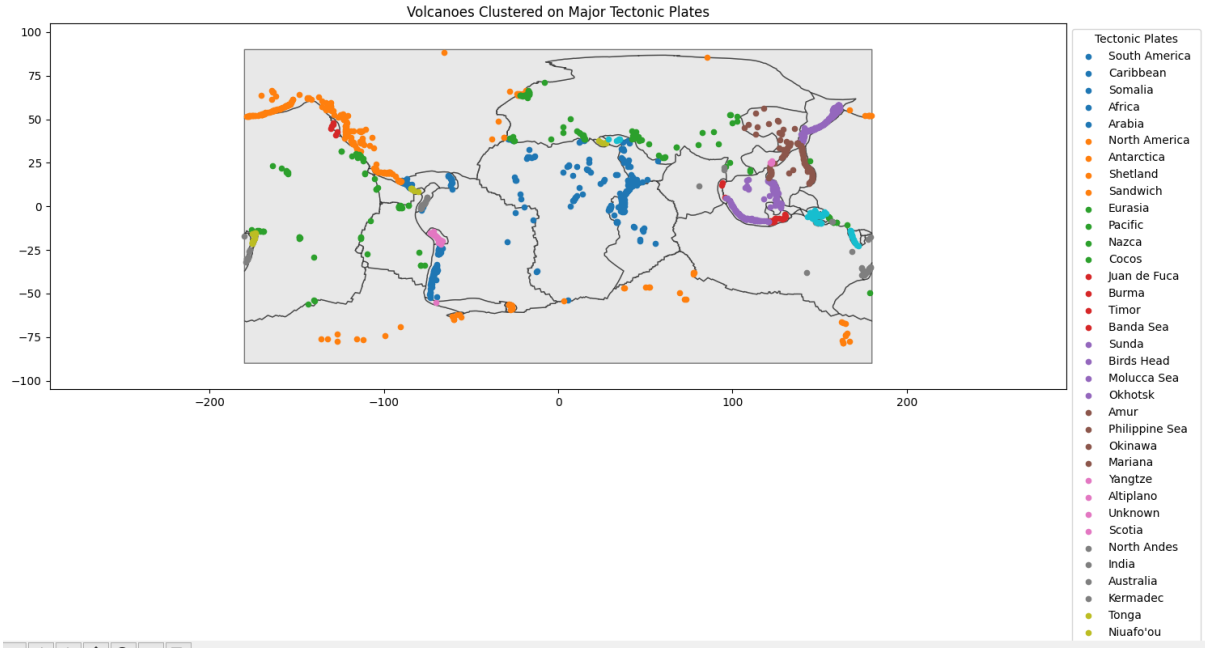
Clustering using only the Major Tectonic plates



Number of volcanoes per Major Tectonic plate

South America: 127
Africa: 84
North America: 238
Antarctica: 33
Eurasia: 80
Pacific: 55
India: 3
Australia: 20

Clustering using all Major, Secondary and Ternary Tectonic Plates



Number of volcanoes per Plate

{'South America': 127, 'Caribbean': 76, 'Somalia': 67, 'Africa': 84, 'Arabia': 49, 'North America': 238, 'Antarctica': 33, 'Shetland': 3, 'Sandwich': 9, 'Eurasia': 80, 'Pacific': 55, 'Nazca': 23, 'Cocos': 1, 'Juan de Fuca': 6, 'Burma': 3, 'Timor': 22, 'Banda Sea': 7, 'Sunda': 143, 'Birds Head': 16, 'Molucca Sea': 1, 'Okhotsk': 213, 'Amur': 31, 'Philippine Sea': 42, 'Okinawa': 14, 'Mariana': 24, 'Yangtze': 4, 'Altiplano': 28, 'Unknown': 1, 'Scotia': 1, 'North Andes': 26, 'India': 3, 'Australia': 20, 'Kermadec': 21, 'Tonga': 17, "Niuafo'ou": 1, 'Panama': 8, 'Aegean Sea': 5, 'Anatolia': 6, 'North Bismarck': 5, 'South Bismarck': 30, 'Woodlark': 11, 'New Hebrides': 17}

Code:

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point

MAJOR_TECTONIC_PLATES = [
    "Pacific", "North America", "South America", "Africa",
    "Eurasia", "India", "Australia", "Antarctica"
]

def init_data():
    volcanoes = pd.read_csv("volcano_data2.csv")
    tectonic_plates = gpd.read_file("PB2002_plates.json")

    return volcanoes, tectonic_plates

def assign_to_plate(volcanoes, plates):
```

```
    plate_names = []
    for volcano in volcanoes.geometry:
        found_plate = "Unknown"
        for _, plate in plates.iterrows():
            if plate['geometry'].contains(volcano):
                found_plate = plate['PlateName']
                break
        plate_names.append(found_plate)

    return plate_names


def plot_volcanoes_onto_plates(volcanoes_df, tectonic_plates, geometry):
    volcanoes_gdf = gpd.GeoDataFrame(volcanoes_df, geometry=geometry)
    volcanoes_gdf['Tectonic_Plate'] = assign_to_plate(volcanoes_gdf,
tectonic_plates)

    _, ax = plt.subplots(figsize=(15, 8))

    tectonic_plates.plot(ax=ax, color="lightgray", edgecolor="black", linewidth=1,
alpha=0.5)
    unique_plates = volcanoes_gdf['Tectonic_Plate'].unique()
    colors = plt.cm.get_cmap('tab10', len(unique_plates))

    for i, plate in enumerate(unique_plates):
        plate_volcanoes = volcanoes_gdf[volcanoes_gdf['Tectonic_Plate'] == plate]
        plate_volcanoes.plot(
            ax=ax,
            color=colors(i),
            markersize=20,
            label=plate
        )

    plt.title("Volcanoes Clustered on Major Tectonic Plates")
    plt.legend(title="Tectonic Plates", loc='upper left', bbox_to_anchor=(1, 1))
    plt.axis('equal')  # Equal scaling for longitude and latitude
    plt.tight_layout()
    plt.show()
    print_volcanoes_per_plate(volcanoes_gdf)

def print_volcanoes_per_plate(volcanoes_gdf):
    plate_volcano_count = {}
    for _, volcano in volcanoes_gdf.iterrows():
        plate = volcano['Tectonic_Plate']
        if plate in plate_volcano_count:
            plate_volcano_count[plate] += 1
        else:
            plate_volcano_count[plate] = 1

    print( 'Total volcanoes per plater: ' + str(plate_volcano_count))
```

```
if __name__ == "__main__":
    volcanoes_df, non_filtered_tectonic_plates = init_data()
    tectonic_plates_major =
non_filtered_tectonic_plates[non_filtered_tectonic_plates['PlateName'].isin(MAJOR_T
ECTONIC_PLATES)]

    geometry = [Point(xy) for xy in zip(volcanoes_df['Longitude'],
volcanoes_df['Latitude'])]

    plot_volcanoes_onto_plates(volcanoes_df, tectonic_plates_major, geometry)
    plot_volcanoes_onto_plates(volcanoes_df, non_filtered_tectonic_plates,
geometry)
```