In [ ]:

```
# ! pip install geopy
```

In [ ]:

```
# pip install geopandas
```

In [1]:

```
# import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import folium
import math
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
import time
import geopandas as gpd
from shapely.geometry import Point
import random
```

In [2]:

```
# pip install --upgrade openpyxl
```

In [3]:

```
# Read the Excel file
df = pd.read_excel('County Location_ Raw Data.xlsx')
```

In [4]:

```
# Check shape of the data
df.shape
```

Out[4]:

```
(570675, 17)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 570675 entries, 0 to 570674
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype         
---  ------              --------------   -----         
 0   Household_ID        570675 non-null  object        
 1   Village_ID          570675 non-null  int64         
 2   Village_Name        570675 non-null  object        
 3   Sublocation_Name    570675 non-null  object        
 4   Sublocation_ID      570675 non-null  int64         
 5   Location_ID         570675 non-null  int64         
 6   Location_Name       570675 non-null  object        
 7   Constituency_Name   570675 non-null  object        
 8   County_Name         570675 non-null  object        
 9   IsBeneficiaryHH     570675 non-null  bool          
 10  latitude            570675 non-null  float64       
 11  longitude           570675 non-null  float64       
 12  RuralUrban          544571 non-null  object        
 13  Constituency_ID     570675 non-null  int64         
 14  Entry_Date          570675 non-null  datetime64[ns]
 15  UserCode            570675 non-null  object        
 16  County_ID           570675 non-null  int64         
dtypes: bool(1), datetime64[ns](1), float64(2), int64(5), object(8)
memory usage: 70.2+ MB
```

In [6]:

```
# Check for non-numeric values
non_numeric_values = df[df['UserCode'].isnull()]

# Print the non-numeric values
print(non_numeric_values)
```

```
Empty DataFrame
Columns: [Household_ID, Village_ID, Village_Name, Sublocation_Name, Sublocation_ID, Location_ID, Locatio
n_Name, Constituency_Name, County_Name, IsBeneficiaryHH, latitude, longitude, RuralUrban, Constituency_I
D, Entry_Date, UserCode, County_ID]
Index: []
```

In [7]:

```
# Check for non-numeric values
non_numeric_values = df[df['Household_ID'].isnull()]

# Print the non-numeric values
print(non_numeric_values)
```

```
Empty DataFrame
Columns: [Household_ID, Village_ID, Village_Name, Sublocation_Name, Sublocation_ID, Location_ID, Locatio
n_Name, Constituency_Name, County_Name, IsBeneficiaryHH, latitude, longitude, RuralUrban, Constituency_I
D, Entry_Date, UserCode, County_ID]
Index: []
```

In [8]:

```
# Convert all object data to title case
df = df.apply(lambda x: x.str.title() if x.dtype == 'object' else x)

# Convert all column names to lower case
df.columns = df.columns.str.lower()
```

In [9]:

```
df.head()
```

Out[9]:

	household_id	village_id	village_name	sublocation_name	sublocation_id	location_id	location_name	const
0	29334629	50902020103141674	El Hache B	Elwak South	509020201	5090202	Elwak South	I
1	4010101010072345100212	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
2	4010101010072345100213	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
3	401010101007234510126	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
4	401010101007234510128	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	

In [10]:

```
# Check for missing values
df.isna().sum()
```

Out[10]:

```
household_id              0
village_id                0
village_name              0
sublocation_name          0
sublocation_id            0
location_id               0
location_name             0
constituency_name         0
county_name               0
isbeneficiaryhh           0
latitude                  0
longitude                 0
ruralurban            26104
constituency_id           0
entry_date                0
usercode                  0
county_id                 0
dtype: int64
```

In [11]:

```
# checkout the counties in the df
df['county_name'].unique()
```

Out[11]:

```
array(['Mandera', 'Marsabit', 'Wajir', 'Turkana'], dtype=object)
```

In [12]:

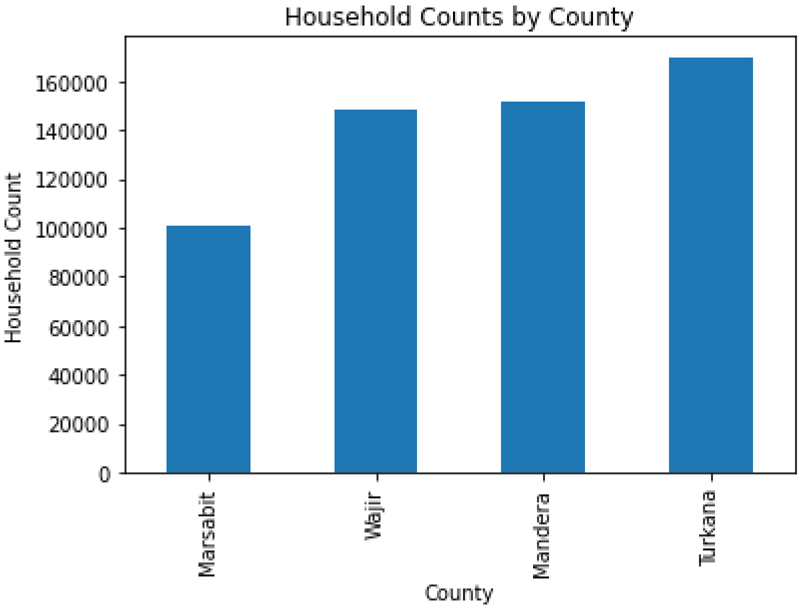
```
# Group the DataFrame by county and count the number of households in each county
county_counts = df.groupby('county_name')['household_id'].count()

# Sort the county counts in ascending order
county_counts = county_counts.sort_values()

# Plot a bar chart of the county counts
county_counts.plot(kind='bar')

# Set the chart title and axis labels
plt.title('Household Counts by County')
plt.xlabel('County')
plt.ylabel('Household Count')

# Show the chart
plt.show()
```



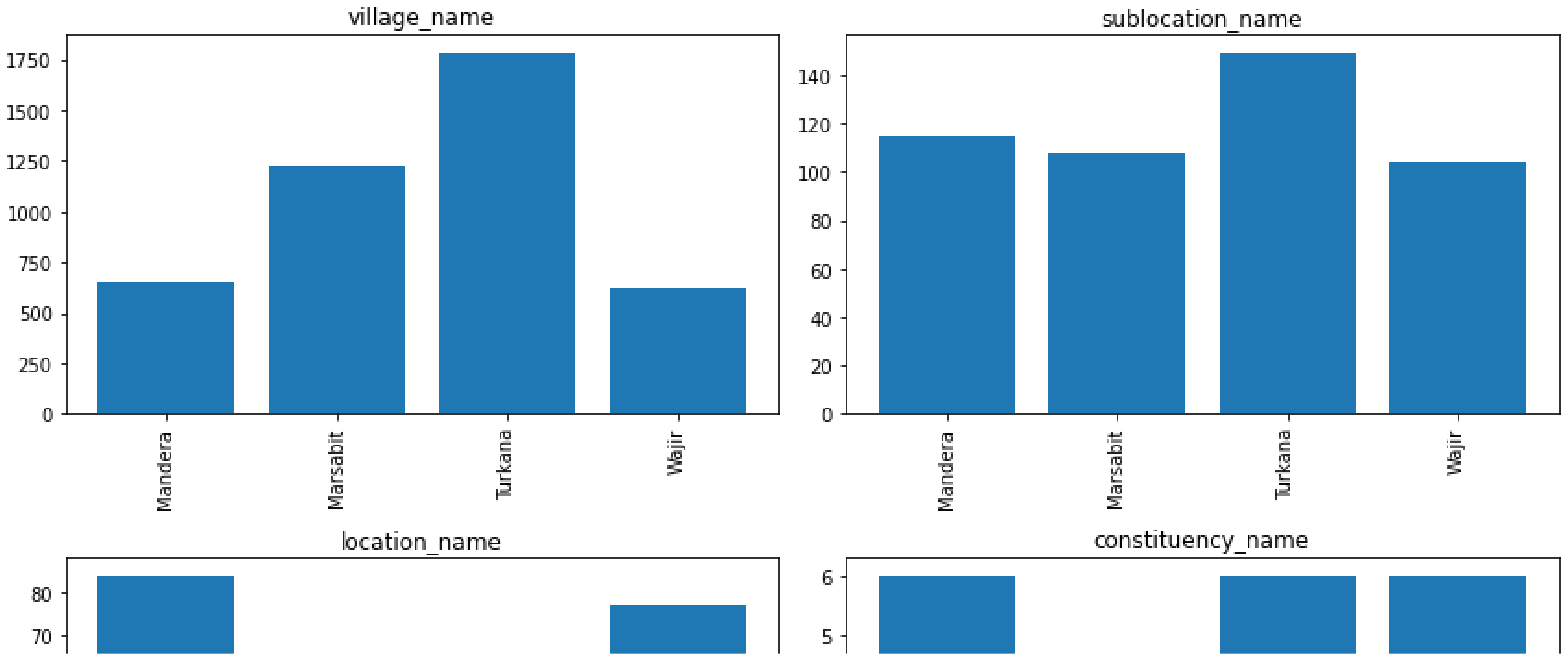
In [13]:

```
# Group the DataFrame by county and count the unique values of Village_Name, Sublocation_Name, Location_Na
county_counts = df.groupby('county_name').nunique()[['village_name', 'sublocation_name', 'location_name',

# Plot four bar graphs, one for each column
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
axes = axes.flatten()

for i, col in enumerate(county_counts.columns):
    ax = axes[i]
    ax.bar(county_counts.index, county_counts[col])
    ax.set_title(col)
    ax.tick_params(axis='x', rotation=90)

plt.tight_layout()
plt.show()
```



1) Marsabit County

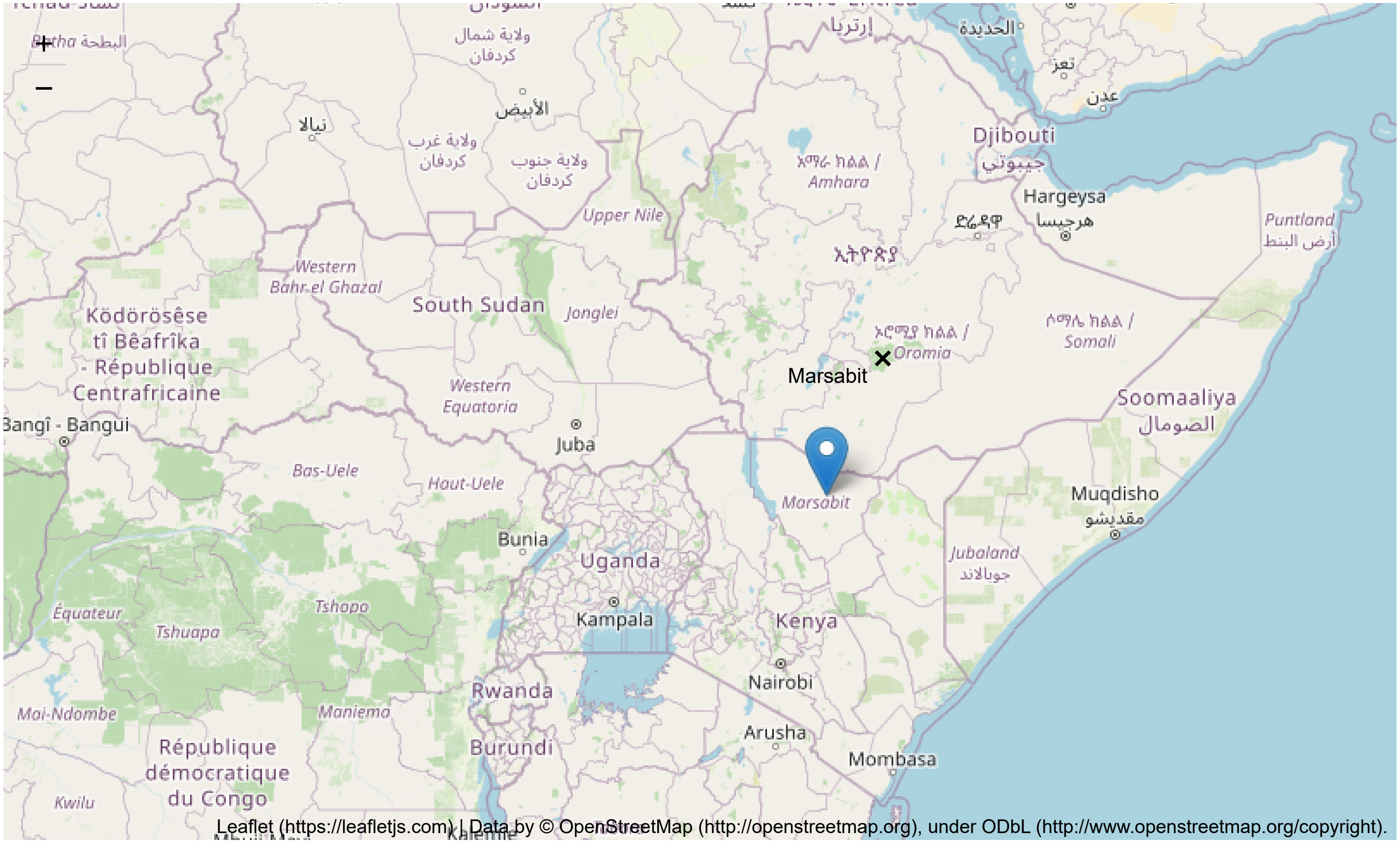
In [14]:

```
# Create a map of Marsabit
marsabit_map = folium.Map(location=[2.96776, 37.98612], zoom_start=5)

# Add a marker for Marsabit town
marsabit_marker = folium.Marker(location=[2.96776, 37.98612], popup='Marsabit')
marsabit_marker.add_to(marsabit_map)

# Show the map
marsabit_map
```

Out[14]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [15]:

```
# Filter out Marsabit in dataframe
mar = df[df['county_name'] == "Marsabit"]
```

In [16]:

```
# data info
mar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100538 entries, 1 to 100538
Data columns (total 17 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   household_id       100538 non-null  object
 1   village_id         100538 non-null  int64
 2   village_name       100538 non-null  object
 3   sublocation_name   100538 non-null  object
 4   sublocation_id     100538 non-null  int64
 5   location_id        100538 non-null  int64
 6   location_name      100538 non-null  object
 7   constituency_name  100538 non-null  object
 8   county_name        100538 non-null  object
 9   isbeneficiaryhh    100538 non-null  bool
 10  latitude           100538 non-null  float64
 11  longitude          100538 non-null  float64
 12  ruralurban         94807 non-null   object
 13  constituency_id    100538 non-null  int64
 14  [illegible]        100538 non-null  datetime64[ns]
```

In [17]:

```
mar.head()
```

Out[17]:

	household_id	village_id	village_name	sublocation_name	sublocation_id	location_id	location_name	constit
1	4010101010072345100212	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
2	4010101010072345100213	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
3	401010101007234510126	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
4	401010101007234510128	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
5	401010101007234510133	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	

In [18]:

```
# shape of dataframe
mar.shape
```

Out[18]:

```
(100538, 17)
```

In [19]:

```
# checkout missing values
mar.isna().any()
```

Out[19]:

```
household_id          False
village_id            False
village_name          False
sublocation_name      False
sublocation_id        False
location_id           False
location_name         False
constituency_name     False
county_name           False
isbeneficiaryhh       False
latitude              False
longitude             False
ruralurban             True
constituency_id       False
entry_date            False
usercode              False
county_id             False
dtype: bool
```

In [20]:

```
# check for duplicates
mar['usercode'].duplicated().sum()
```

Out[20]:

```
100472
```

In [21]:

```
categorical_columns = ["household_id", "village_name", "sublocation_name", "location_name", "constituency_

for column in categorical_columns:
    mar[column] = mar[column].str.strip()
```

```
<ipython-input-21-0a8e629ac144>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  mar[column] = mar[column].str.strip()
```

In [22]:

```
mar.head()
```

Out[22]:

	household_id	village_id	village_name	sublocation_name	sublocation_id	location_id	location_name	constit
1	4010101010072345100212	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
2	4010101010072345100213	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
3	401010101007234510126	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
4	401010101007234510128	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
5	401010101007234510133	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	

In [23]:

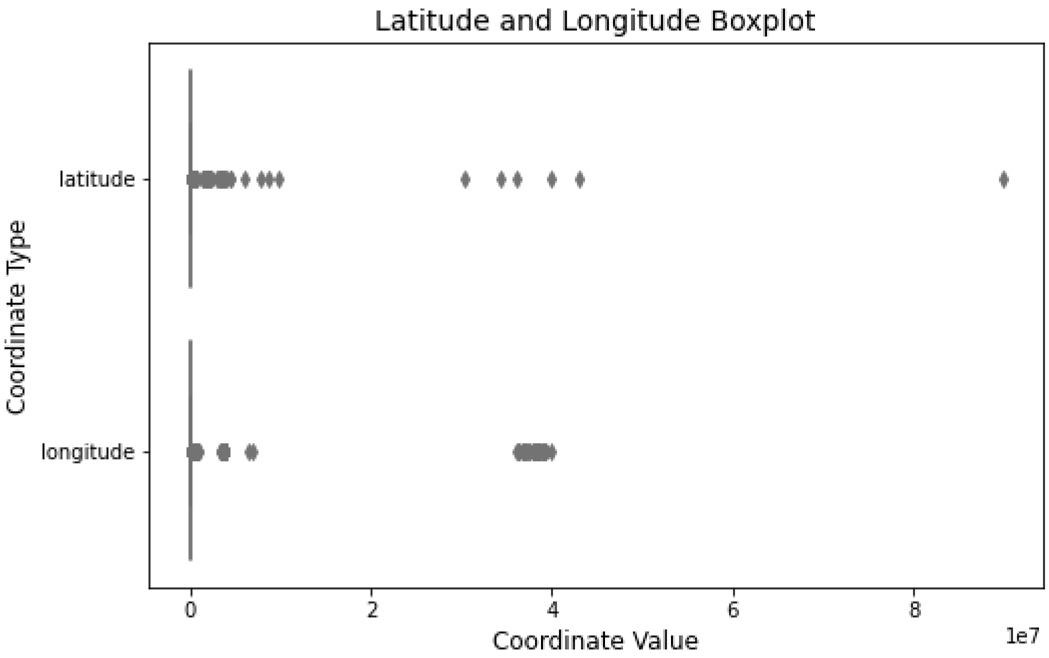
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [24]:

```
# If latitude is greater than longitude they interchange
mask = mar['latitude'] > mar['longitude']
mar.loc[mask, ['latitude', 'longitude']] = mar.loc[mask, ['longitude', 'latitude']].values
```

In [25]:

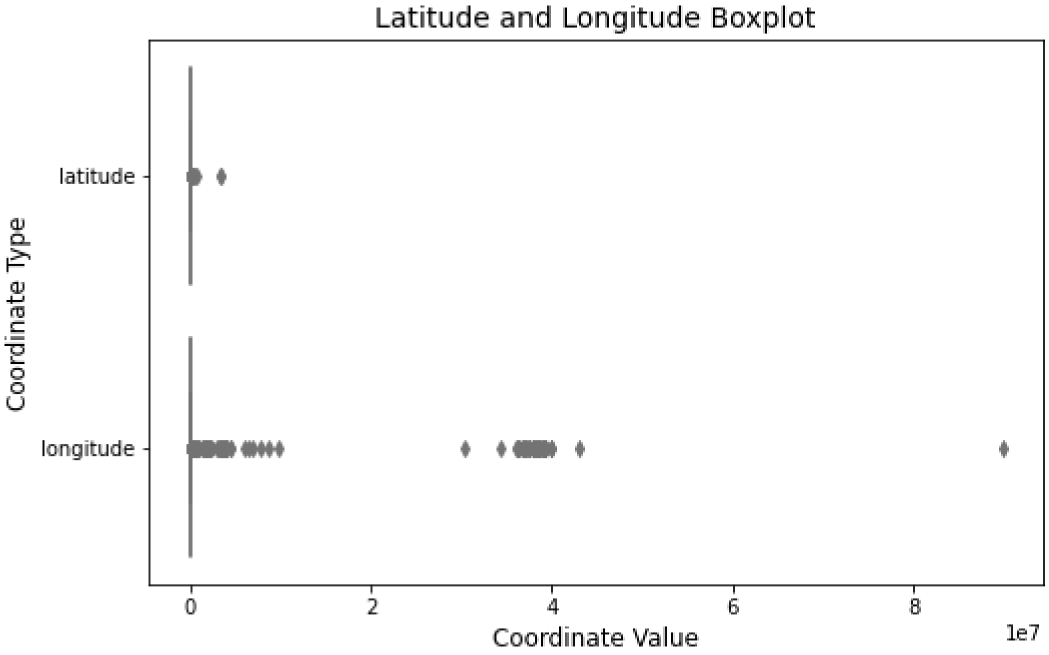
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [26]:

```
print("Min lat = ",mar['latitude'].min(), "Max lat = ",mar['latitude'].max())
```

```
Min lat =  0.0 Max lat =  3333333.0
```

In [27]:

```
print("Min long = ",mar['longitude'].min(), "Max long = ",mar['longitude'].max())
```

```
Min long =  0.0 Max long =  89823455.0
```

In [28]:

```
# calculate the correction factor for each value
factors = mar['latitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 1))

# divide each value by its correction factor
mar['latitude'] = mar['latitude'] * factors
```

```
<ipython-input-28-33719cec4d01>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  mar['latitude'] = mar['latitude'] * factors
```

In [29]:

```
print("Min lat = ",mar['latitude'].min(), "Max lat = ",mar['latitude'].max())
```

```
Min lat =  0.0 Max lat =  9.9
```

In [30]:

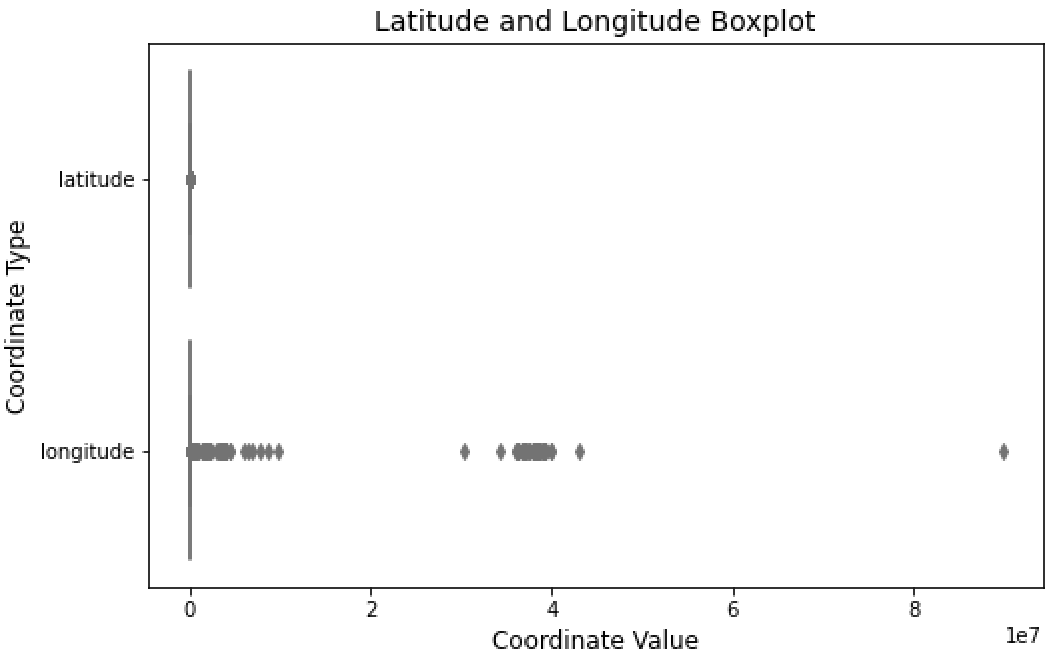
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [31]:

```
# calculate the correction factor for each value
factors = mar['longitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 2))

# divide each value by its correction factor
mar['longitude'] = mar['longitude'] * factors
```

```
<ipython-input-31-7ef8bf162a08>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  mar['longitude'] = mar['longitude'] * factors
```

In [32]:

```
print("Min long = ",mar['longitude'].min(), "Max long = ",mar['longitude'].max())
```

```
Min long =  0.0 Max long =  99.999
```

In [33]:

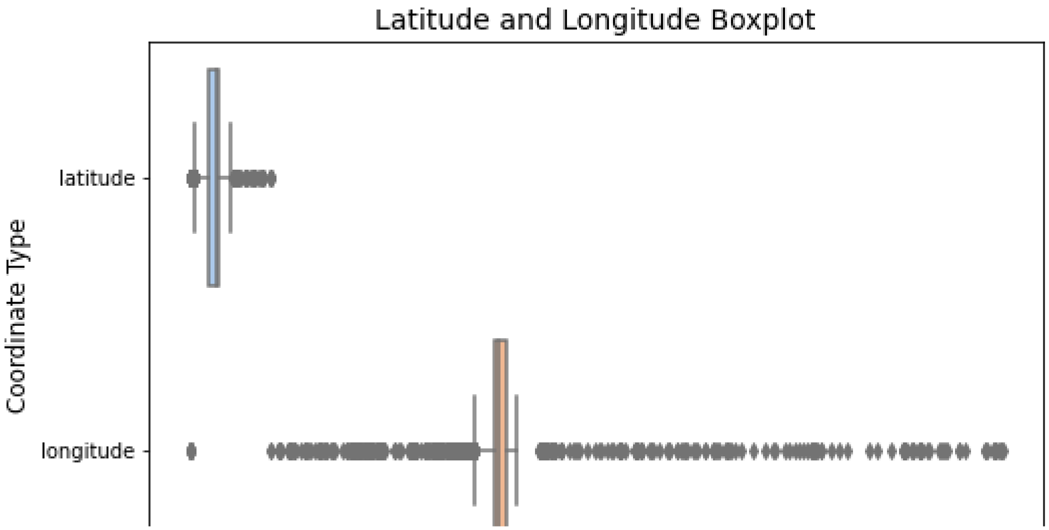
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: ite
ritems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [34]:

```
# Calculate the mean latitude and longitude
mean_latitude = mar['latitude'].mean()
mean_longitude = mar['longitude'].mean()
```

In [35]:

```
mean_latitude
```

Out[35]:

```
3.006505675082725
```

In [36]:

```
mean_longitude
```

Out[36]:

```
38.10805760671885
```

In [37]:

```
# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 10 * mar['latitude'].std()
```

In [38]:

```
threshold
```

Out[38]:

```
7.394863080445737
```

In [39]:

```
# Identify outliers based on the threshold
outliers = ((mar['latitude'] - mean_latitude).abs() > threshold) | ((mar['longitude'] - mean_longitude).ab
```

In [40]:

```
# Replace outliers with random values around the mean
mar.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mar['latitude'].std(), size=outl
mar.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mar['longitude'].std(), size=c
```

In [41]:

```
mar.head()
```

Out[41]:

```
            household_id         village_id  village_name  sublocation_name  sublocation_id  location_id  location_name  constit
1  4010101010072345100212  4010101010072345  Galcha Dida          Nyayo Rd       401010101      4010101         Nagayo
2  4010101010072345100213  4010101010072345  Galcha Dida          Nyayo Rd       401010101      4010101         Nagayo
3   401010101007234510126  4010101010072345  Galcha Dida          Nyayo Rd       401010101      4010101         Nagayo
4   401010101007234510128  4010101010072345  Galcha Dida          Nyayo Rd       401010101      4010101         Nagayo
5   401010101007234510133  4010101010072345  Galcha Dida          Nyayo Rd       401010101      4010101         Nagayo
```

In [42]:

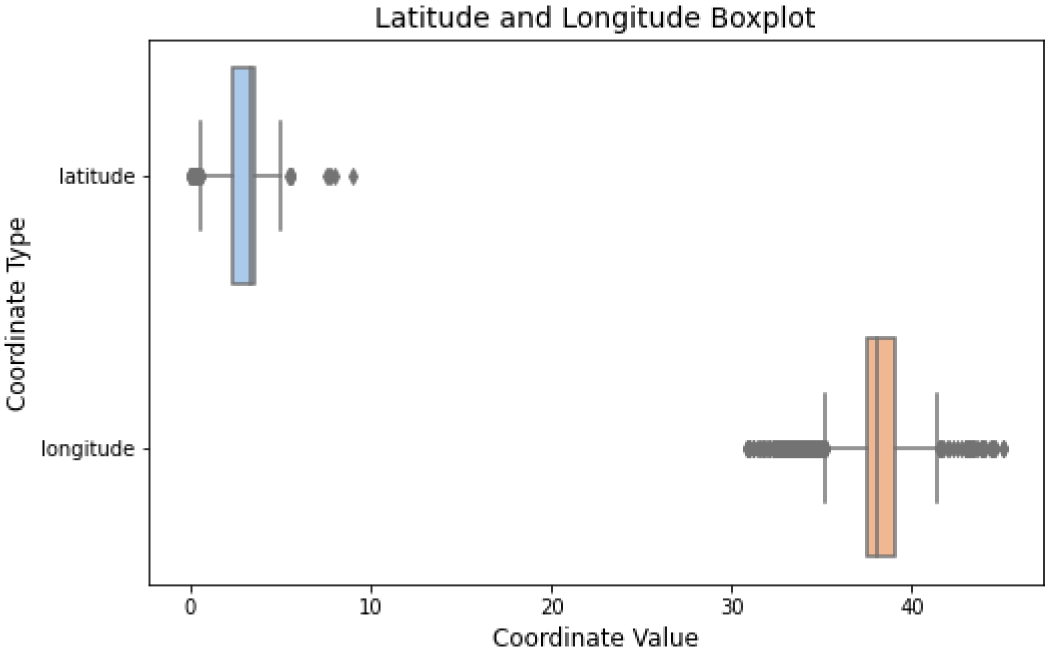
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [43]:

```
# Calculate the mean latitude and longitude
mean_latitude = mar['latitude'].mean()
mean_longitude = mar['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mar['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mar['latitude'] - mean_latitude).abs() > threshold) | ((mar['longitude'] - mean_longitude).ab

# Replace outliers with random values around the mean
mar.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mar['latitude'].std(), size=outl
mar.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mar['longitude'].std(), size=c
```

In [44]:

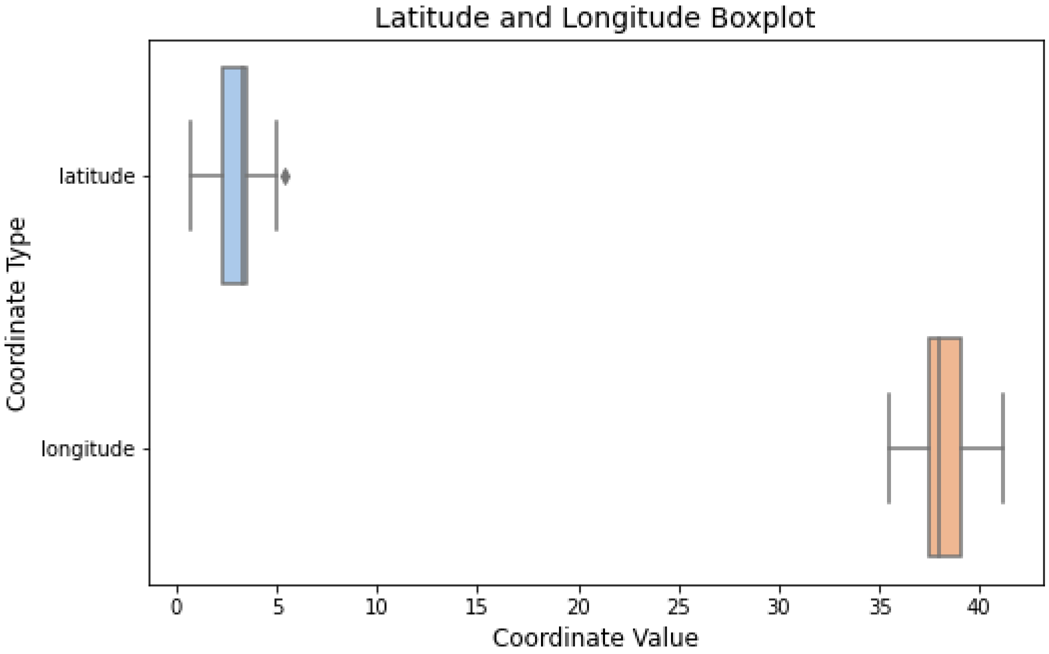
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [45]:

```
# Calculate the mean latitude and longitude
mean_latitude = mar['latitude'].mean()
mean_longitude = mar['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mar['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mar['latitude'] - mean_latitude).abs() > threshold) | ((mar['longitude'] - mean_longitude).ab

# Replace outliers with random values around the mean
mar.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mar['latitude'].std(), size=outl
mar.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mar['longitude'].std(), size=c
```

In [46]:

```
# Calculate the mean latitude and longitude
mean_latitude = mar['latitude'].mean()
mean_longitude = mar['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mar['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mar['latitude'] - mean_latitude).abs() > threshold) | ((mar['longitude'] - mean_longitude).ab

# Replace outliers with random values around the mean
mar.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mar['latitude'].std(), size=outl
mar.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mar['longitude'].std(), size=c
```

In [47]:

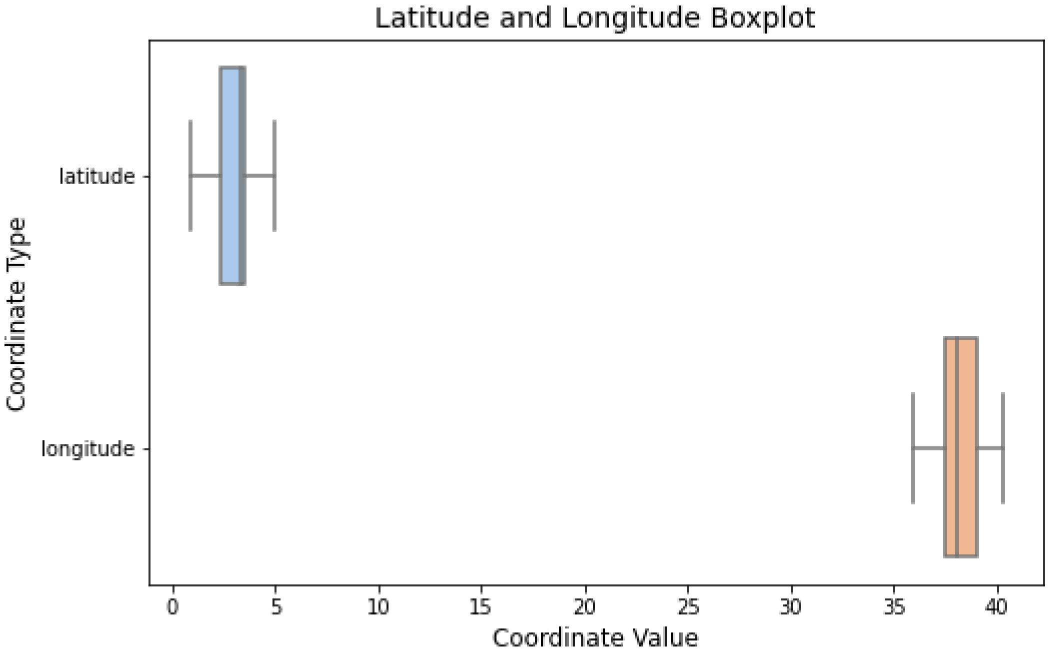
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mar[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [48]:

```
# Read the Marsabit boundary shapefile into a GeoDataFrame
marsabit_boundary = gpd.read_file('Shapefiles/Marsabit_County.shp')  # Replace 'marsabit_boundary.shp' wit
```

In [49]:

```
# Function to check if a coordinate is in Marsabit
def is_coordinate_in_marsabit(latitude, longitude):
    point = Point(longitude, latitude)
    return marsabit_boundary.contains(point).any()

# Iterate through the dataframe rows and update coordinates if necessary
for index, row in mar.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']

    if not is_coordinate_in_marsabit(latitude, longitude):
        # Assign random coordinates within Marsabit
        while True:
            random_latitude = random.uniform(2.2769, 4.6429)  # Set the latitude range to cover the approx
            random_longitude = random.uniform(36.4842, 40.9980 )  # Set the longitude range to cover the a

            if is_coordinate_in_marsabit(random_latitude, random_longitude):
                # Found a random coordinate within Marsabit, update the dataframe
                mar.at[index, 'latitude'] = random_latitude
                mar.at[index, 'longitude'] = random_longitude
                break
```

In [ ]:

```
# # #  save dataset
mar.to_csv('marsabit.csv', index=False)
```

2) Wajir

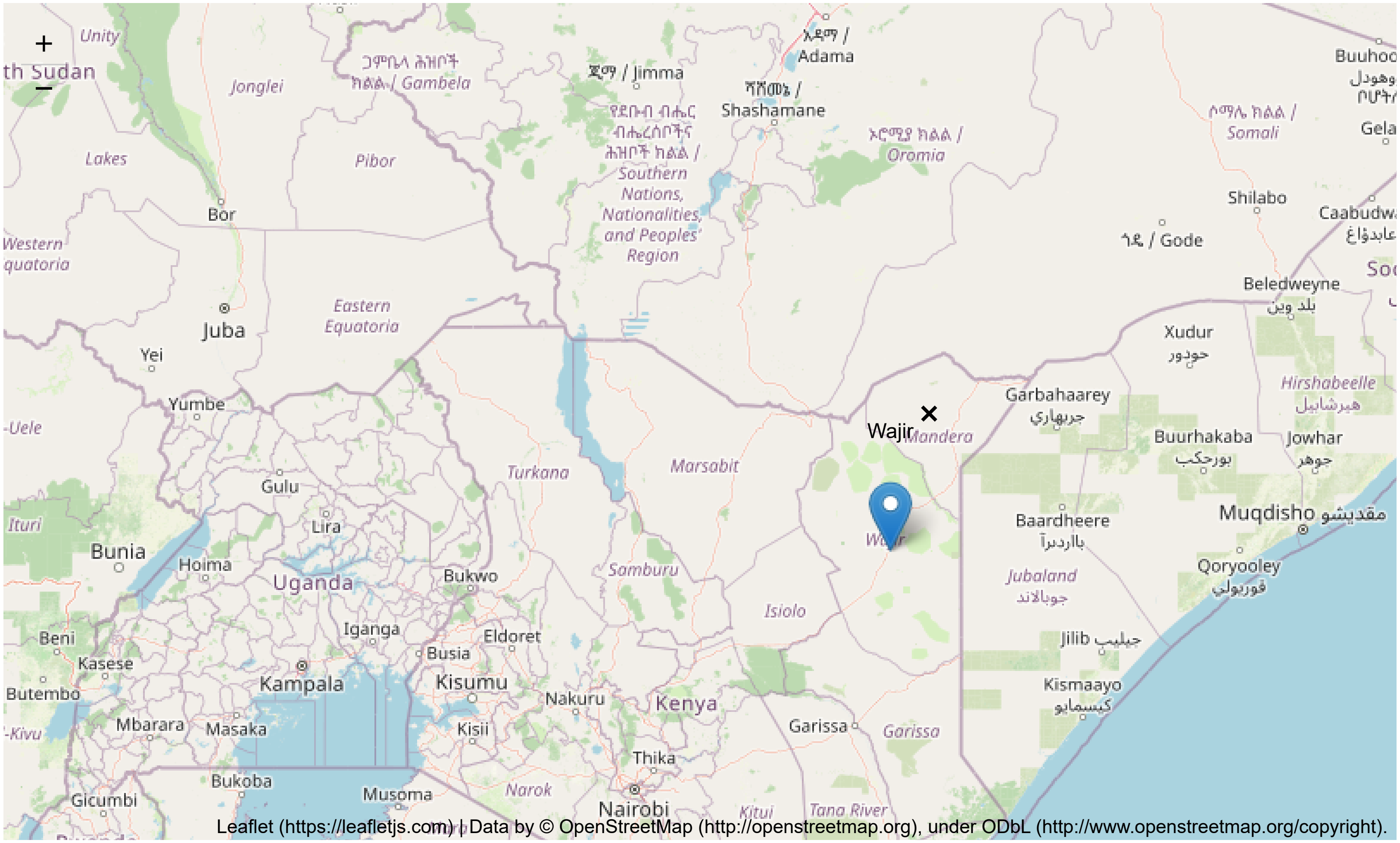
In [76]:

```
# Create a map of Marsabit
wajir_map = folium.Map(location=[1.7470, 40.0682], zoom_start=10)

# Add a marker for Marsabit town
wajir_marker = folium.Marker(location=[1.7470, 40.0682], popup='Wajir')
wajir_marker.add_to(wajir_map)

# Show the map
wajir_map
```

Out[76]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [77]:

```
# Filter out Marsabit in dataframe
wajir = df[df['county_name'] == "Wajir"]
```

In [78]:

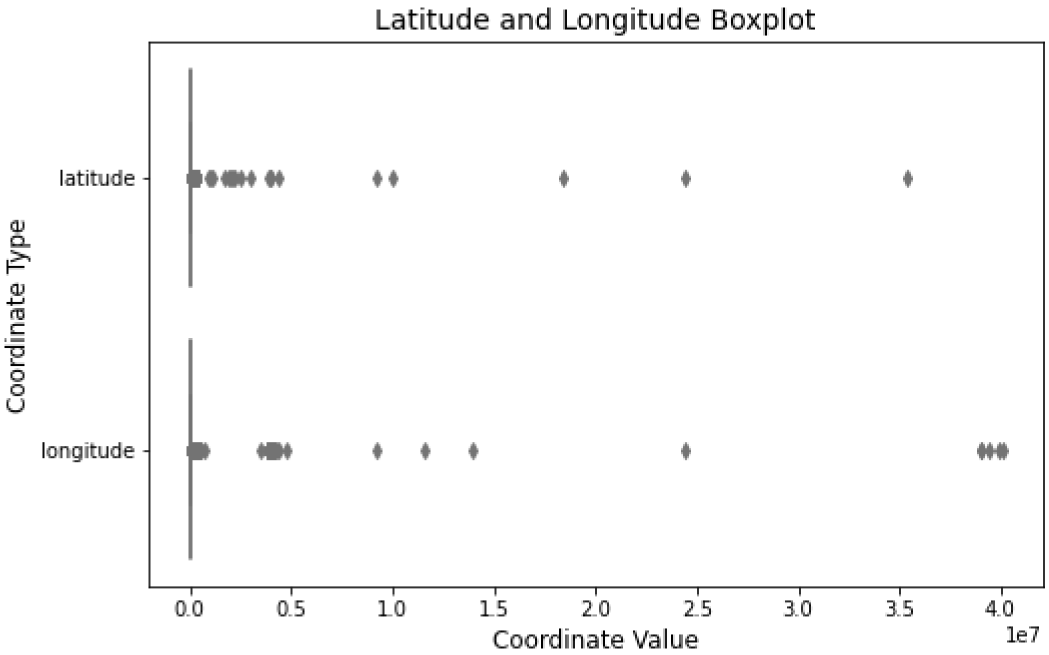
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [79]:

```
# If latitude is greater than longitude they interchange
mask = wajir['latitude'] > wajir['longitude']
wajir.loc[mask, ['latitude', 'longitude']] = wajir.loc[mask, ['longitude', 'latitude']].values
```

In [80]:

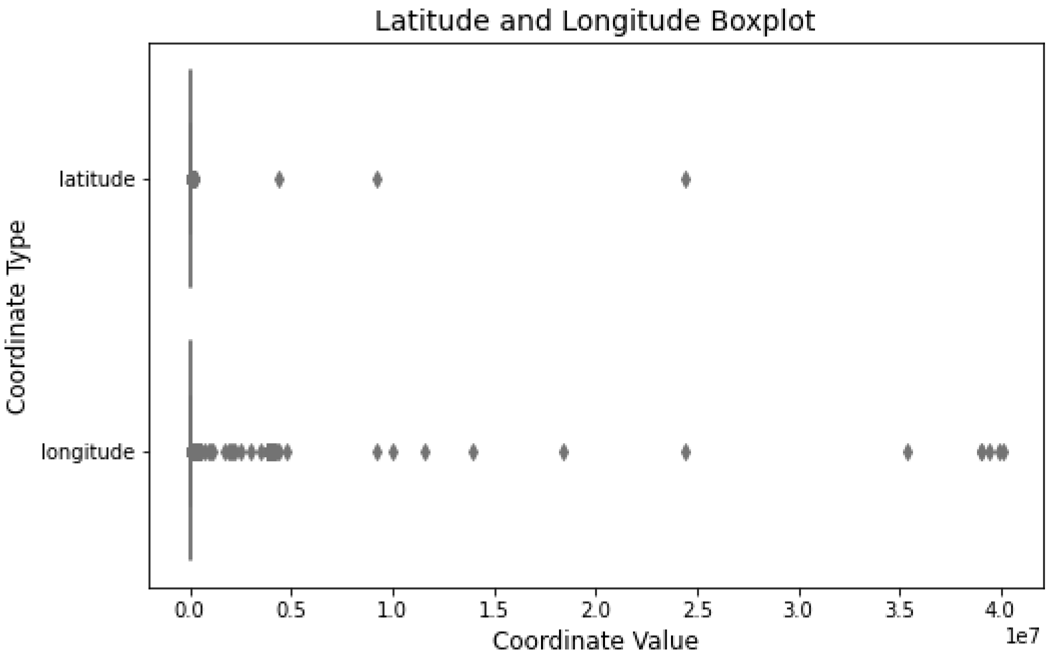
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [81]:

```
# calculate the correction factor for each value
factors = wajir['latitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 1))

# divide each value by its correction factor
wajir['latitude'] = wajir['latitude'] * factors

# # print the corrected dataframe
# print(df_sample)
```

```
<ipython-input-81-0be54d746f88>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  wajir['latitude'] = wajir['latitude'] * factors
```

In [82]:

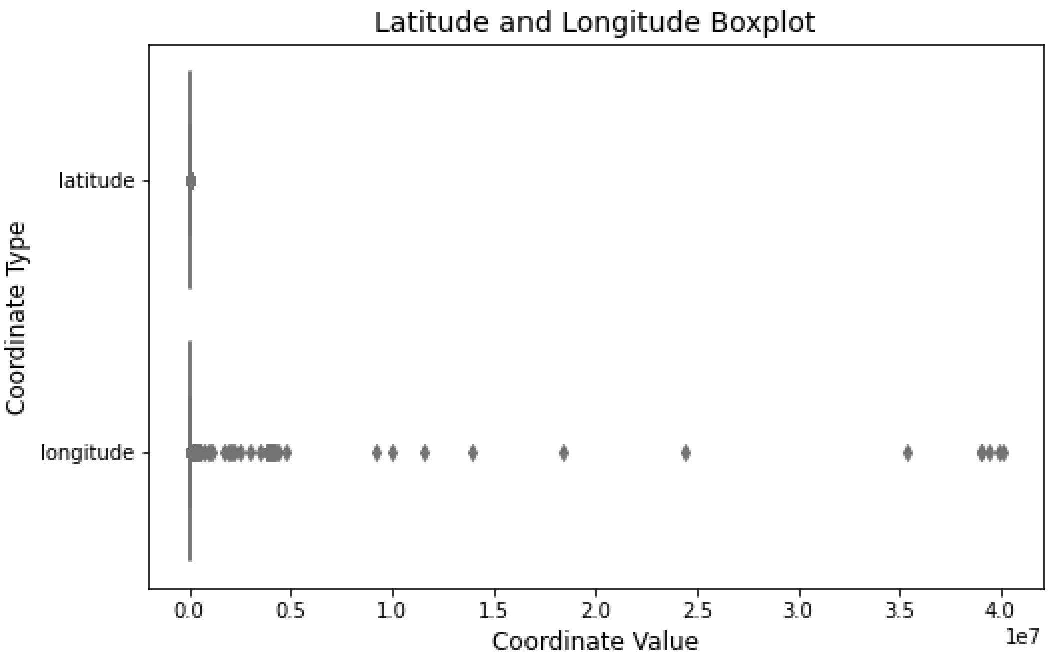
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [83]:

```
# calculate the correction factor for each value
factors = wajir['longitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 2))

# divide each value by its correction factor
wajir['longitude'] = wajir['longitude'] * factors
```

```
<ipython-input-83-b91090bf27e7>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  wajir['longitude'] = wajir['longitude'] * factors
```

In [84]:

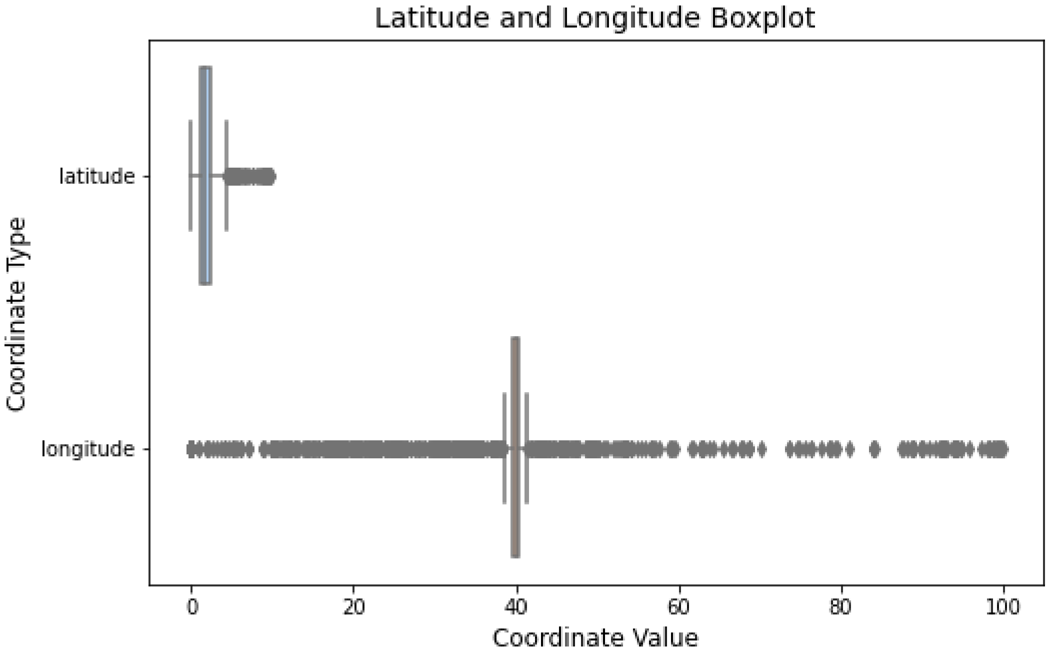
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [85]:

```
# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 10 * wajir['latitude'].std()
```

In [86]:

```
# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude
```

In [87]:

```
# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [88]:

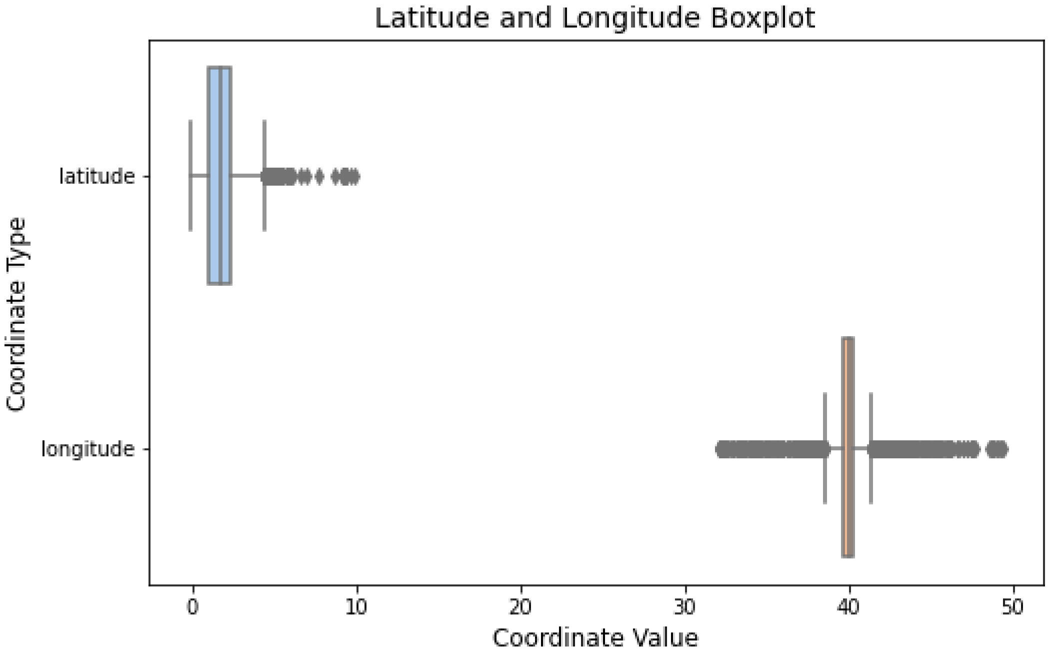
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [89]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [90]:

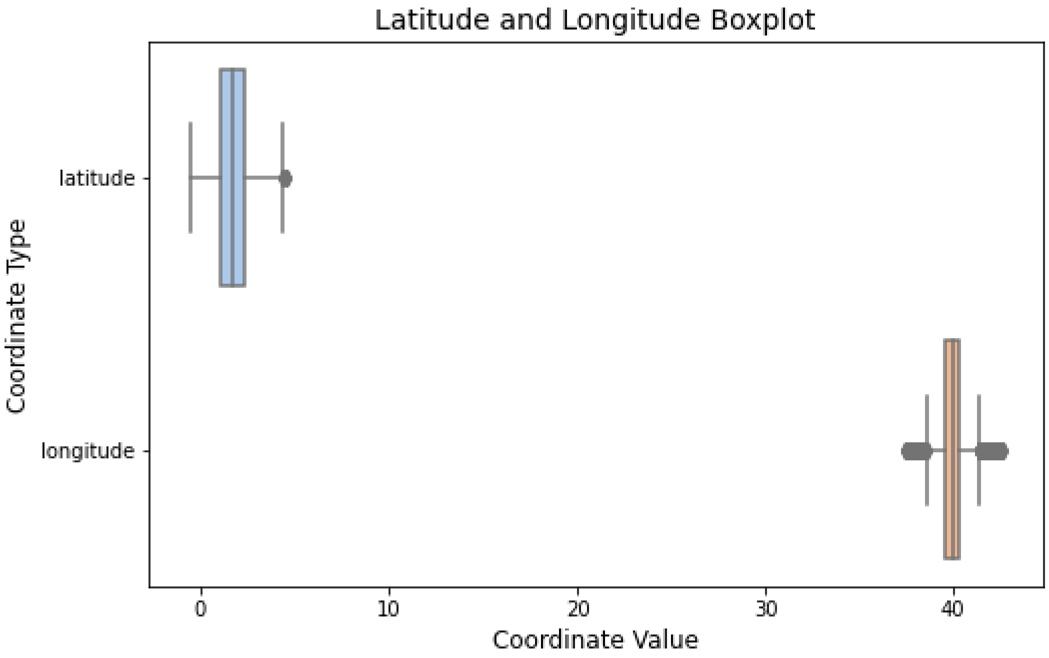
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [91]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [92]:

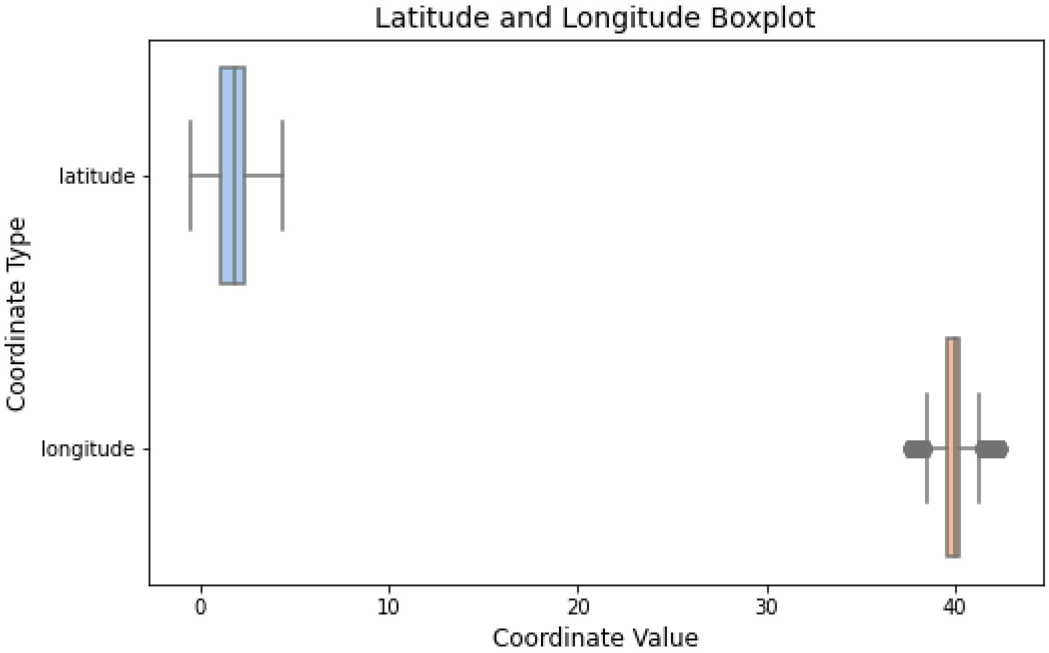
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [93]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [94]:

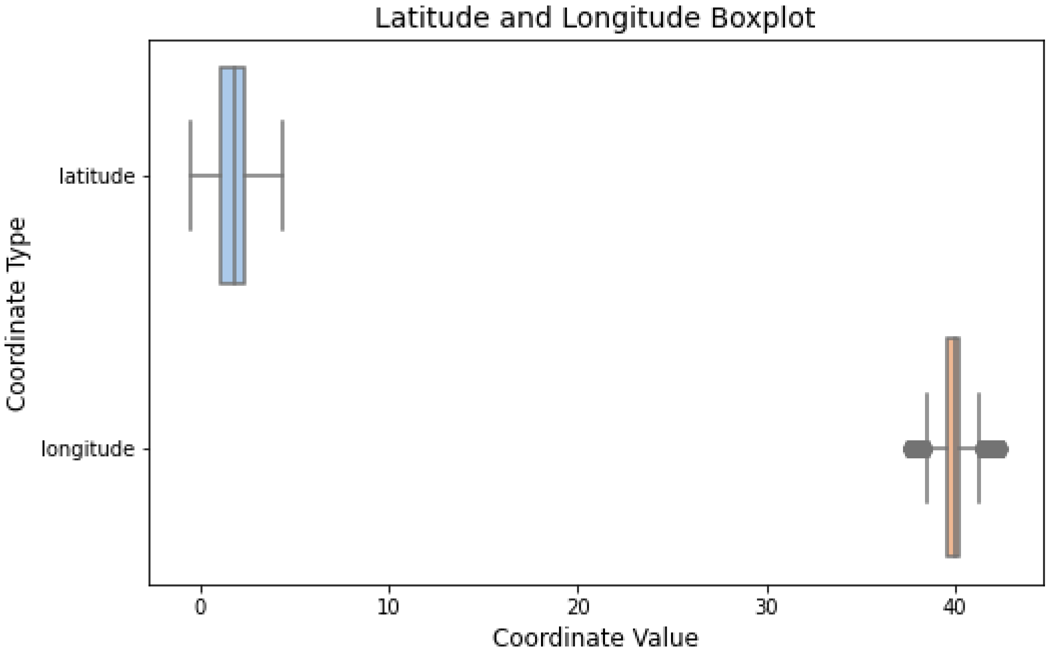
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [95]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [96]:

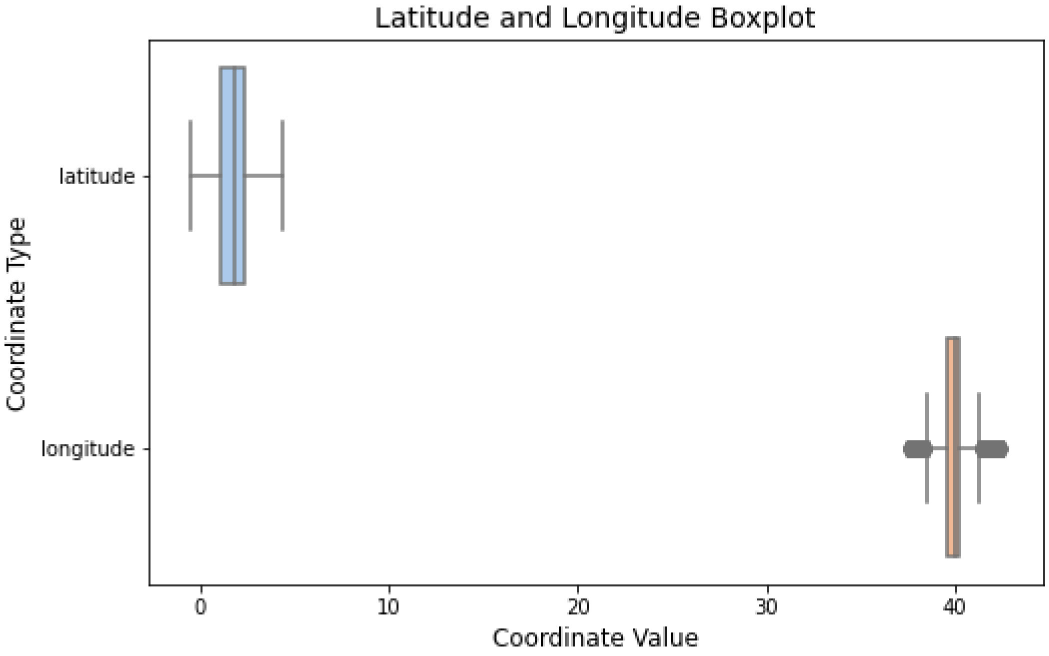
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [97]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold =3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [98]:

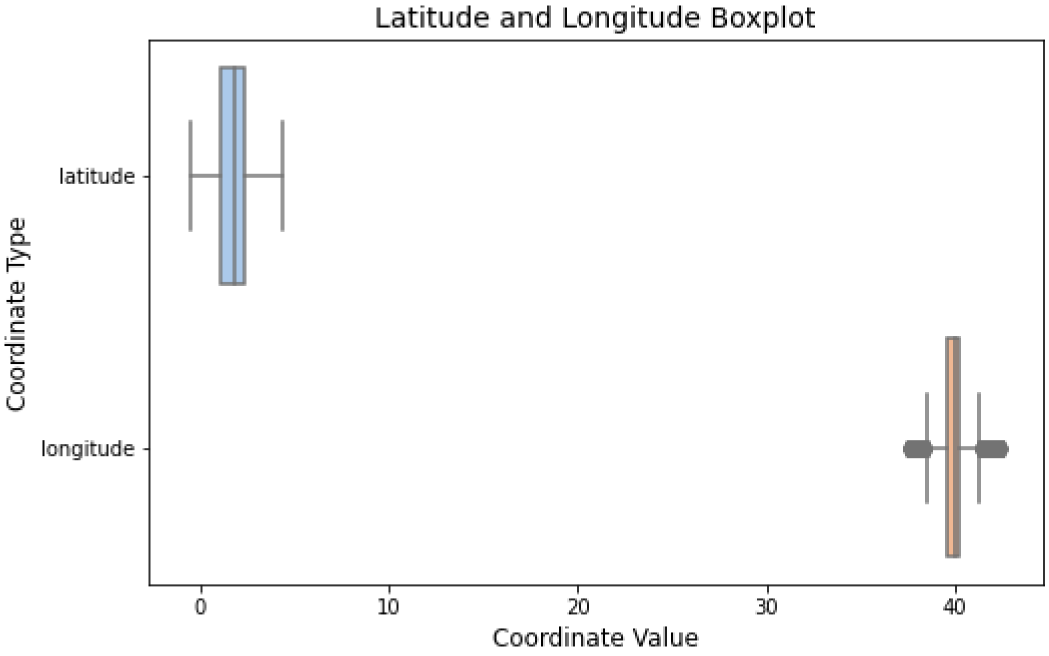
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [99]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [100]:

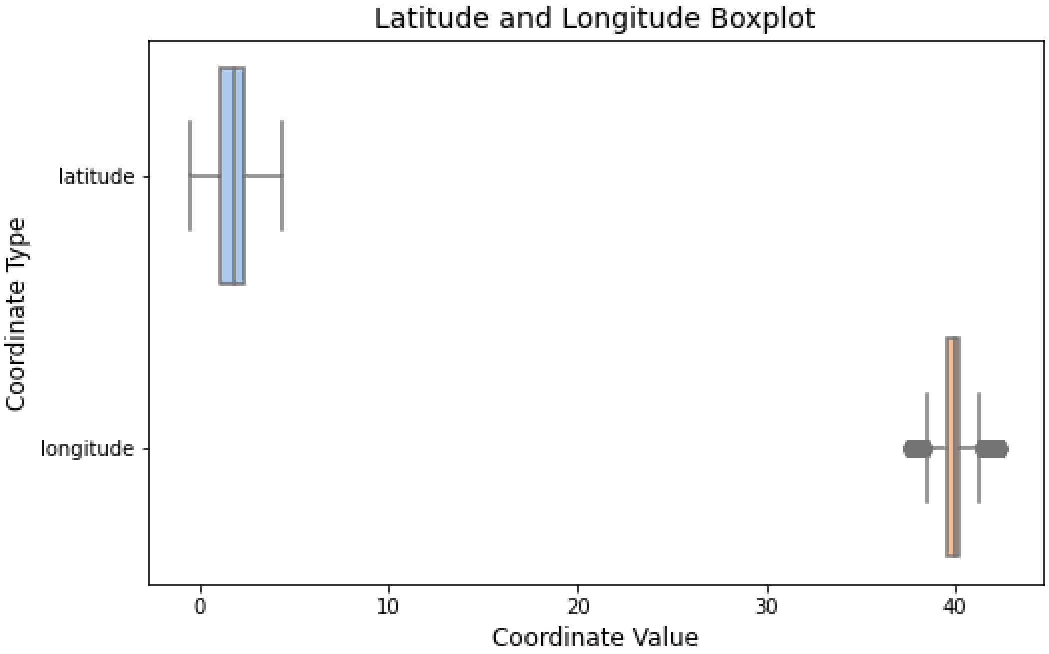
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [101]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [102]:

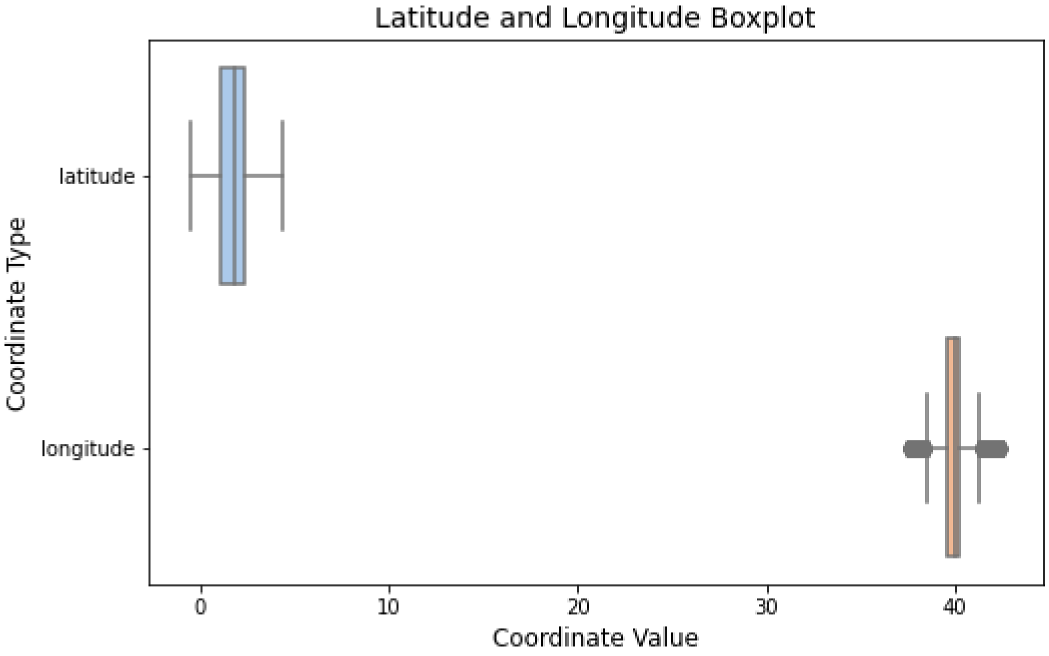
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [103]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [104]:

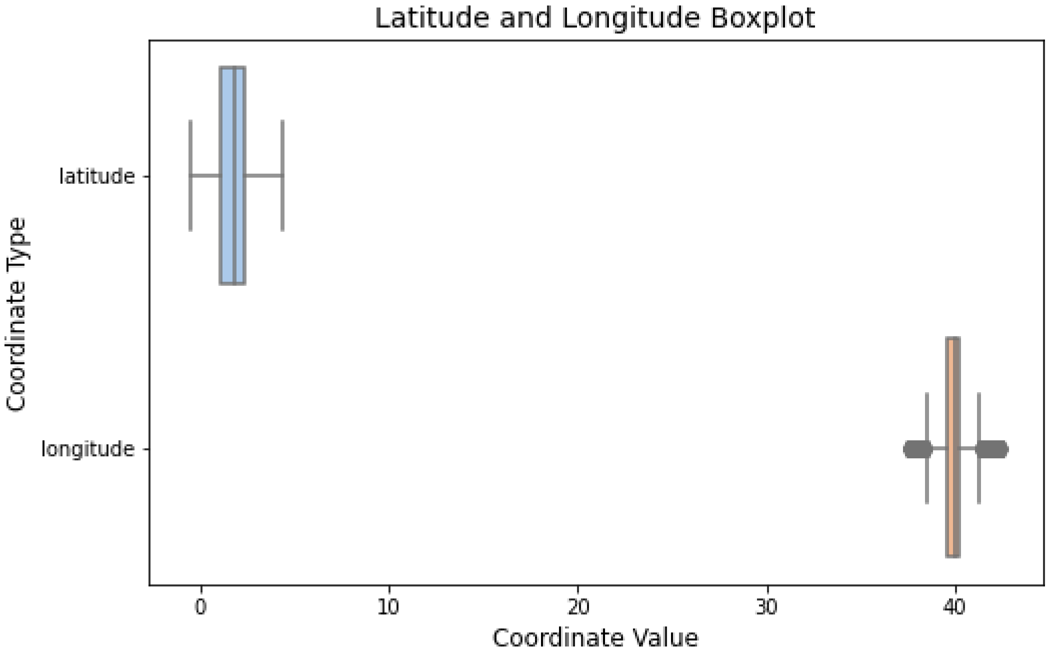
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [105]:

```
# Calculate the mean latitude and longitude
mean_latitude = wajir['latitude'].mean()
mean_longitude = wajir['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * wajir['latitude'].std()

# Identify outliers based on the threshold
outliers = ((wajir['latitude'] - mean_latitude).abs() > threshold) | ((wajir['longitude'] - mean_longitude

# Replace outliers with random values around the mean
wajir.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=wajir['latitude'].std(), size=
wajir.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=wajir['longitude'].std(), si
```

In [106]:

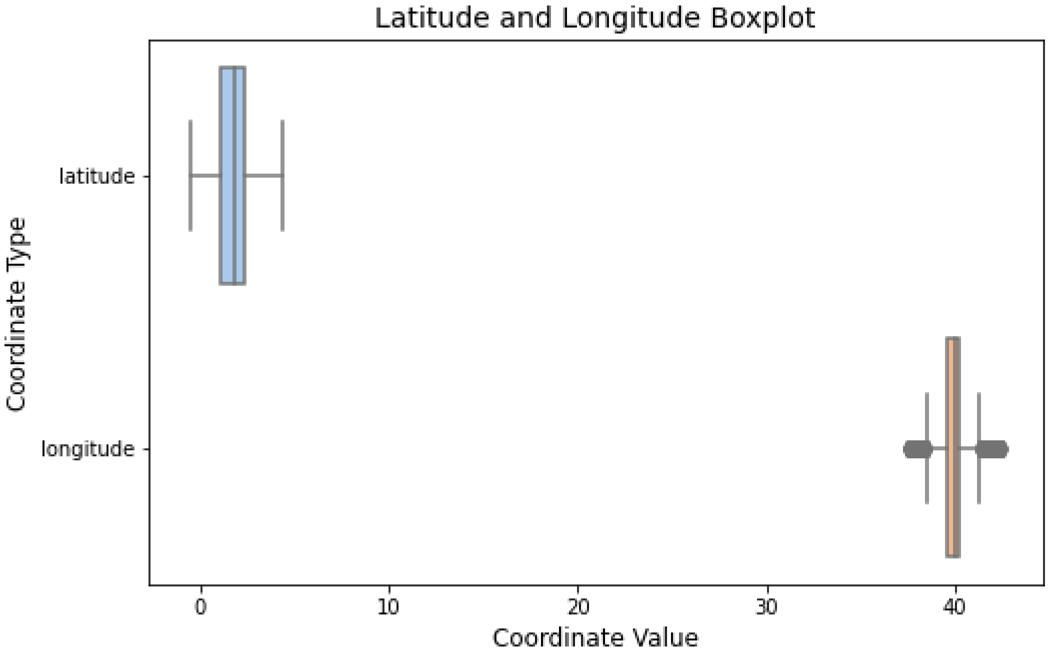
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=wajir[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [107]:

```
# Read the Wajir boundary shapefile into a GeoDataFrame
wajir_boundary = gpd.read_file('Shapefiles/Wajir_County.shp')  # Replace 'wajir_boundary.shp' with the act
```

In [108]:

```
# Function to check if a coordinate is in Wajir
def is_coordinate_in_wajir(latitude, longitude):
    point = Point(longitude, latitude)
    return wajir_boundary.contains(point).any()

# Iterate through the dataframe rows and update coordinates if necessary
for index, row in wajir.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']

    if not is_coordinate_in_wajir(latitude, longitude):
        # Assign random coordinates within Wajir
        while True:
            random_latitude = random.uniform(wajir_boundary.bounds['miny'], wajir_boundary.bounds['maxy'])
            random_longitude = random.uniform(wajir_boundary.bounds['minx'], wajir_boundary.bounds['maxx']

            if is_coordinate_in_wajir(random_latitude, random_longitude):
                # Found a random coordinate within Wajir, update the dataframe
                wajir.at[index, 'latitude'] = random_latitude
                wajir.at[index, 'longitude'] = random_longitude
                break
```

In [109]:

```
## save dataset
wajir.to_csv('wajir.csv', index=False)
```

3) Mandera

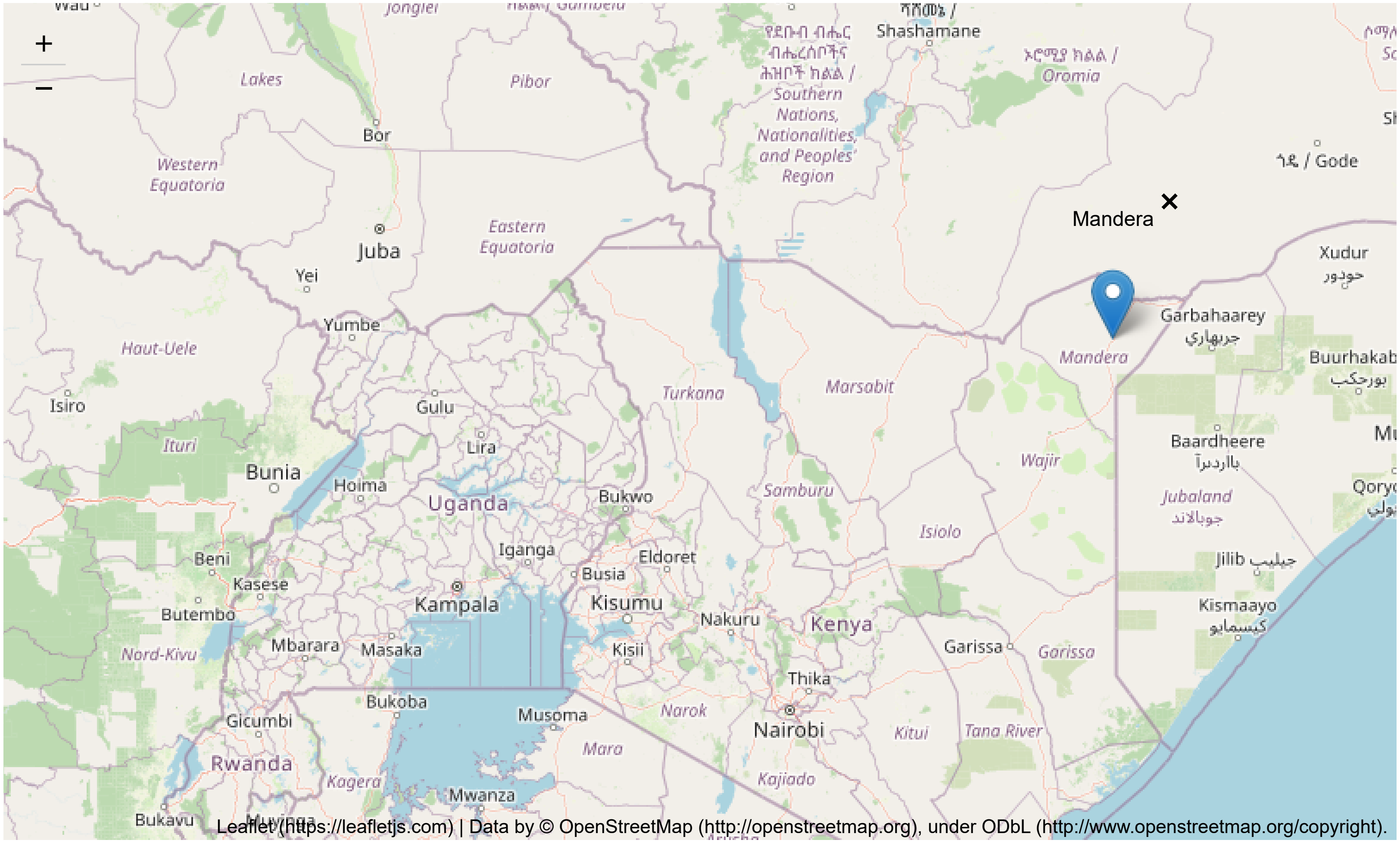
In [110]:

```
# Create a map of Mandera
mandera_map = folium.Map(location=[3.4421, 40.9382], zoom_start=10)

# Add a marker for Marsabit town
mandera_marker = folium.Marker(location=[3.4421, 40.9382], popup='Mandera')
mandera_marker.add_to(mandera_map)

# Show the map
mandera_map
```

Out[110]:



In [111]:

```
# Filter out Marsabit in dataframe
mandera = df[df['county_name'] == "Mandera"]
```

In [112]:

```
mandera.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 151606 entries, 0 to 400783
Data columns (total 17 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   household_id       151606 non-null  object
 1   village_id         151606 non-null  int64
 2   village_name       151606 non-null  object
 3   sublocation_name   151606 non-null  object
 4   sublocation_id     151606 non-null  int64
 5   location_id        151606 non-null  int64
 6   location_name      151606 non-null  object
 7   constituency_name  151606 non-null  object
 8   county_name        151606 non-null  object
 9   isbeneficiaryhh    151606 non-null  bool
 10  latitude           151606 non-null  float64
 11  longitude          151606 non-null  float64
 12  ruralurban         139321 non-null  object
 13  constituency_id    151606 non-null  int64
 14  entry_date         151606 non-null  datetime64[ns]
 15  usercode           151606 non-null  object
 16  county_id          151606 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(2), int64(5), object(8)
memory usage: 19.8+ MB
```

In [113]:

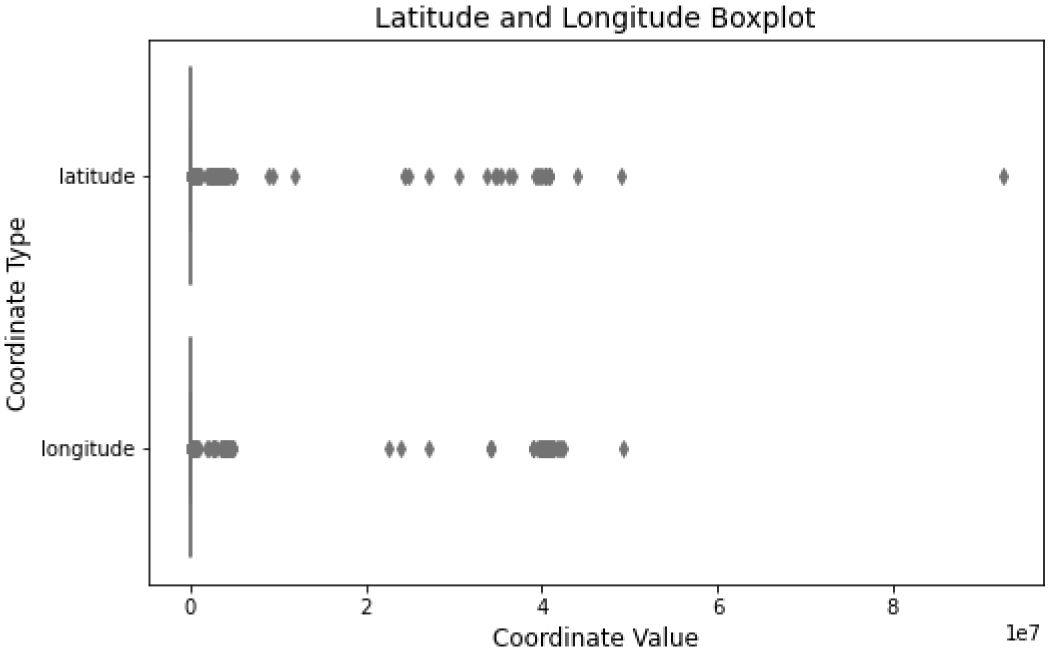
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [114]:

```
# If latitude is greater than longitude they interchange
mask = mandera['latitude'] > mandera['longitude']
mandera.loc[mask, ['latitude', 'longitude']] = mandera.loc[mask, ['longitude', 'latitude']].values
```

In [115]:

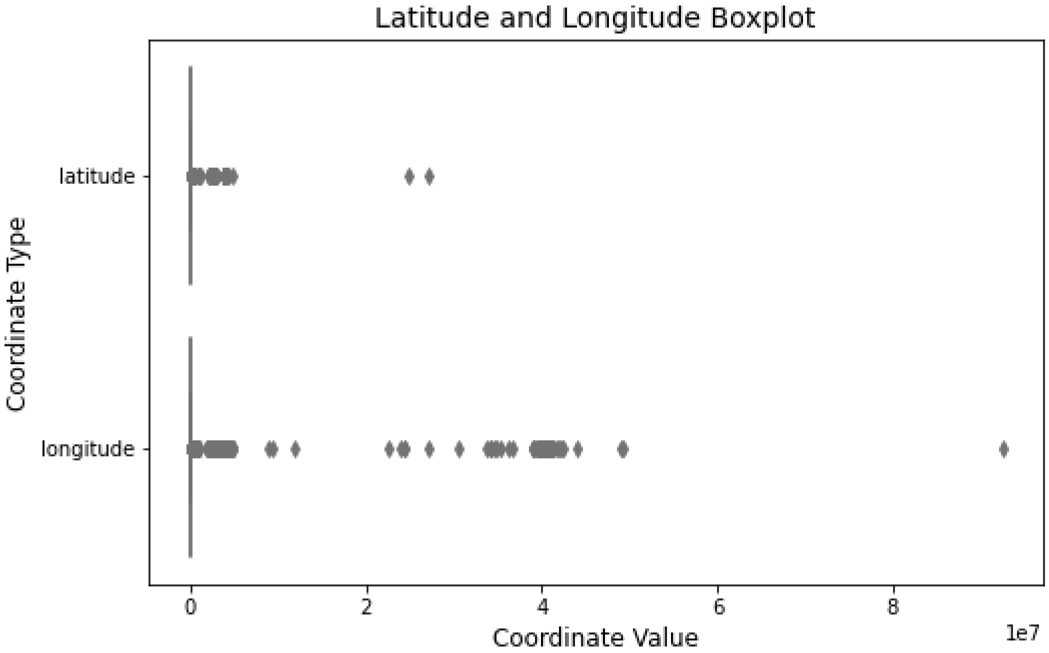
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [116]:

```
# calculate the correction factor for each value
factors = mandera['latitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 1))

# divide each value by its correction factor
mandera['latitude'] = mandera['latitude'] * factors

# # print the corrected dataframe
# print(df_sample)
```

```
<ipython-input-116-92c563b375b9>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  mandera['latitude'] = mandera['latitude'] * factors
```

In [117]:

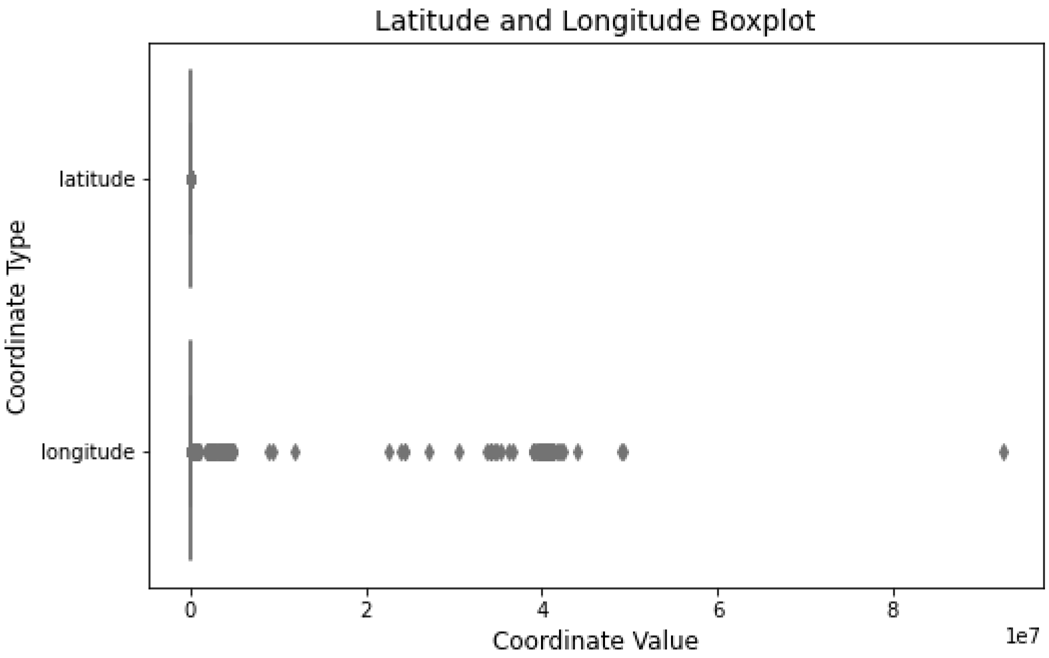
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [118]:

```python
# calculate the correction factor for each value
factors = mandera['longitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 2))

# divide each value by its correction factor
mandera['longitude'] = mandera['longitude'] * factors
```

```
<ipython-input-118-9ee05d1aeb9e>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  mandera['longitude'] = mandera['longitude'] * factors
```

In [119]:

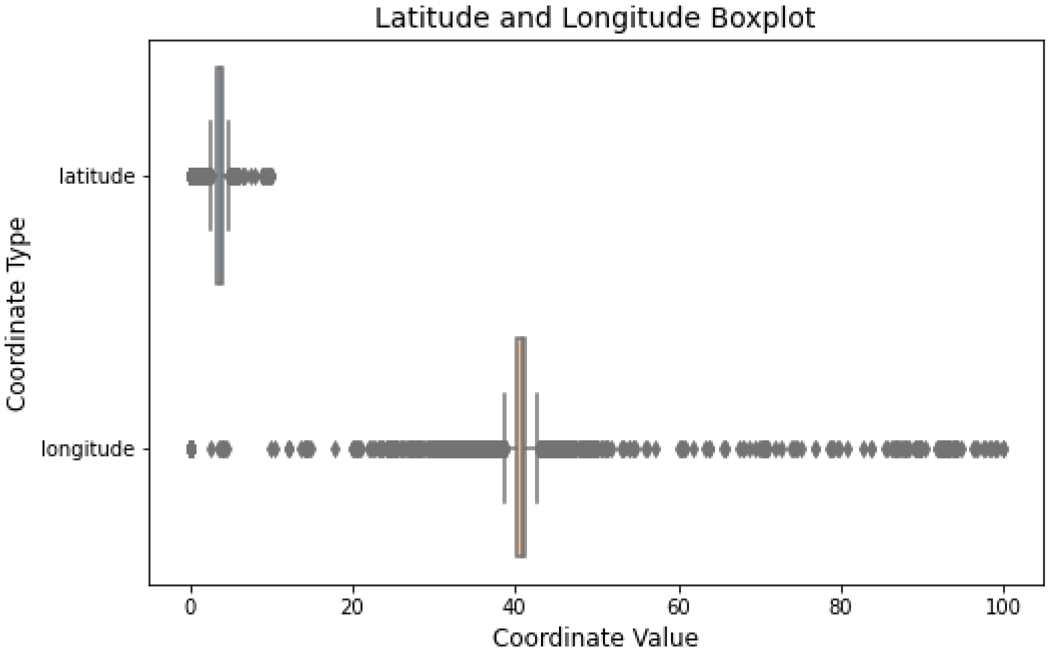
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [120]:

```
# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 10 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [121]:

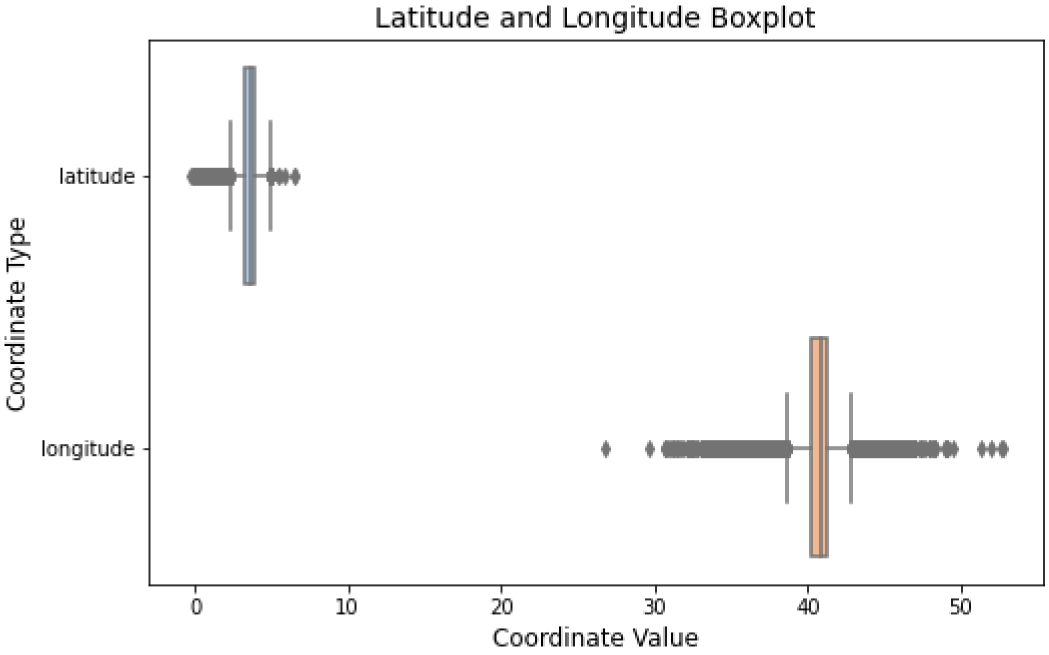
```python
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [122]:

```
# Calculate the mean latitude and longitude
mean_latitude = mandera['latitude'].mean()
mean_longitude = mandera['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [123]:

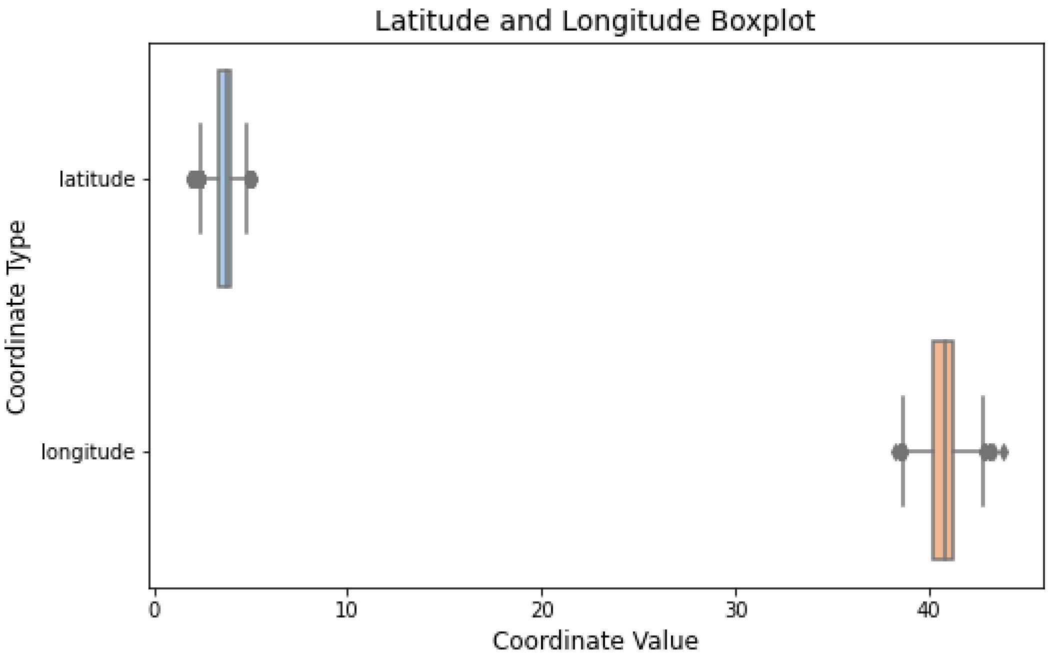
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [124]:

```
# Calculate the mean latitude and longitude
mean_latitude = mandera['latitude'].mean()
mean_longitude = mandera['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [125]:

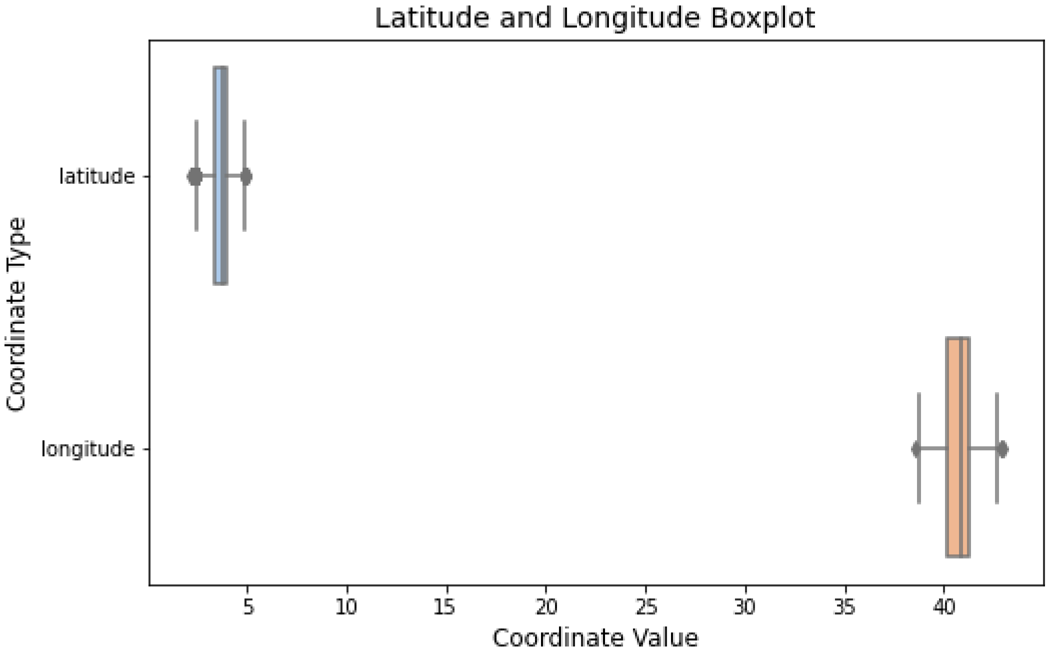
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [127]:

```
# Calculate the mean latitude and longitude
mean_latitude = mandera['latitude'].mean()
mean_longitude = mandera['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [128]:

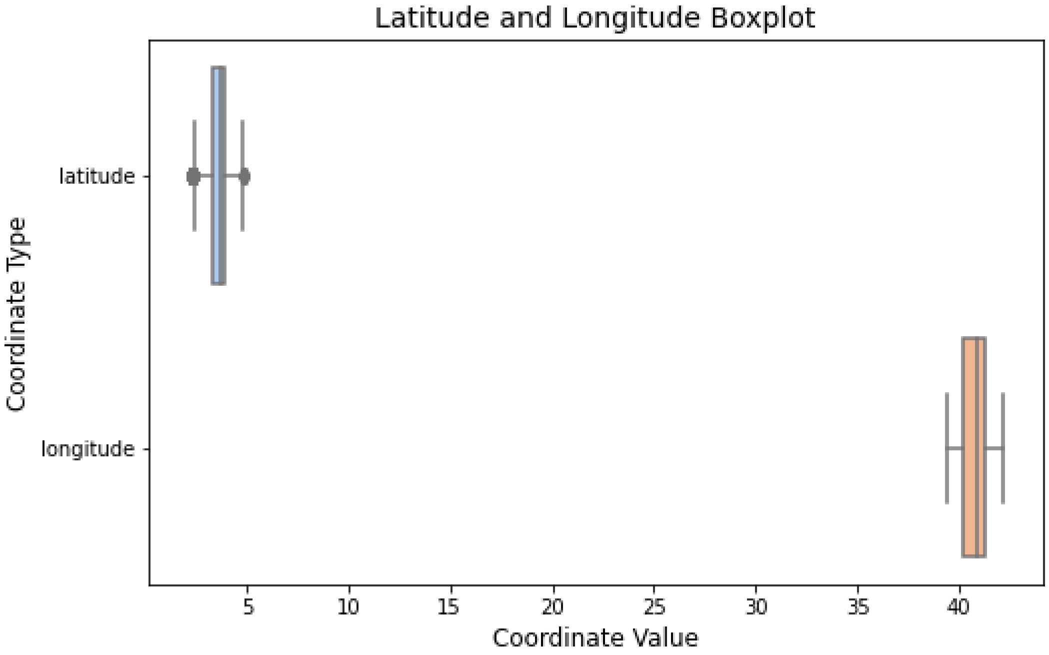
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [129]:

```
# Calculate the mean latitude and longitude
mean_latitude = mandera['latitude'].mean()
mean_longitude = mandera['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [130]:

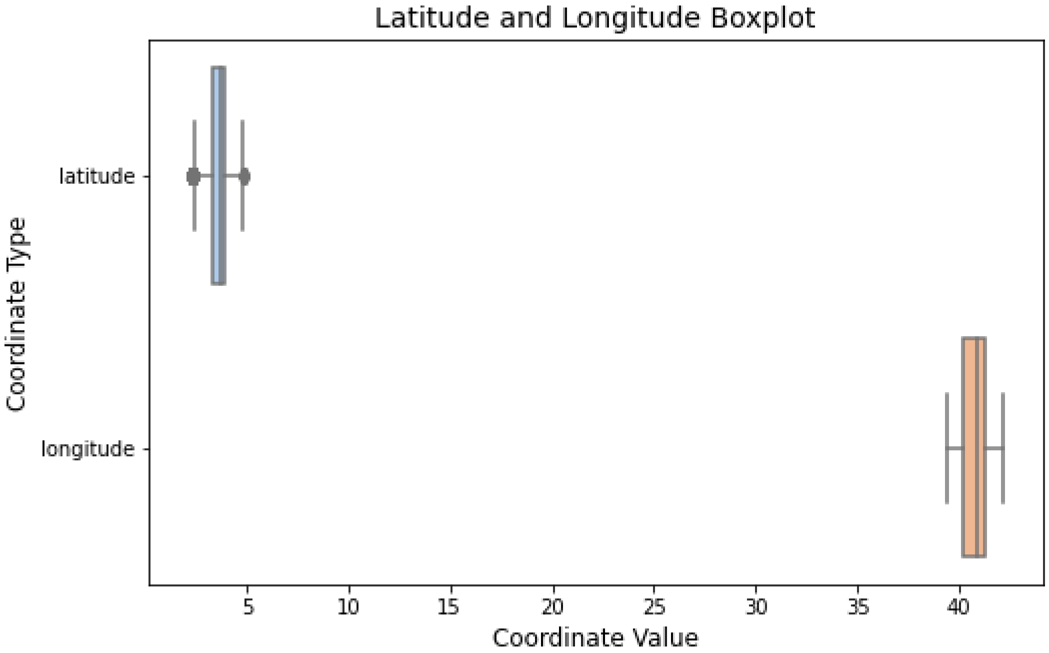
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [131]:

```
# Calculate the mean latitude and longitude
mean_latitude = mandera['latitude'].mean()
mean_longitude = mandera['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * mandera['latitude'].std()

# Identify outliers based on the threshold
outliers = ((mandera['latitude'] - mean_latitude).abs() > threshold) | ((mandera['longitude'] - mean_longi

# Replace outliers with random values around the mean
mandera.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=mandera['latitude'].std(), s
mandera.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=mandera['longitude'].std()
```

In [132]:

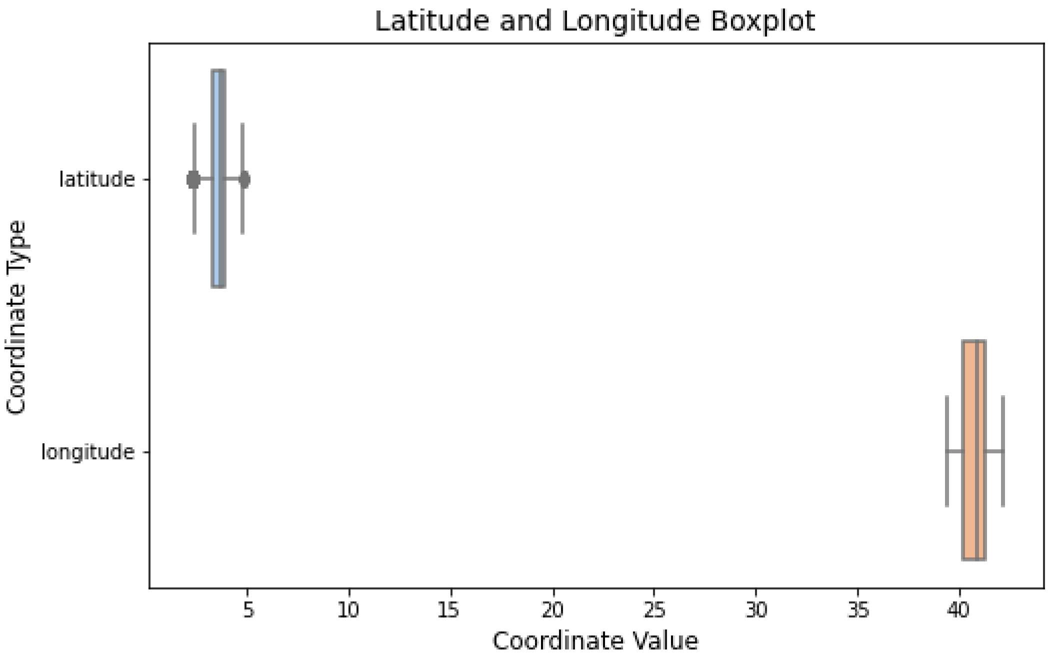
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=mandera[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [133]:

```
# Read the Mandera boundary shapefile into a GeoDataFrame
mandera_boundary = gpd.read_file('Shapefiles/Mandera_County.shp')  # Replace 'mandera_boundary.shp' with t
```

In [134]:

```
# Function to check if a coordinate is in Marsabit
def is_coordinate_in_mandera(latitude, longitude):
    point = Point(longitude, latitude)
    return mandera_boundary.contains(point).any()

# Iterate through the dataframe rows and update coordinates if necessary
for index, row in mandera.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']

    if not is_coordinate_in_mandera(latitude, longitude):
        # Assign random coordinates within Mandera
        while True:
            # Set the latitude range to cover the approximate area of Mandera
            random_latitude = random.uniform(mandera_boundary.bounds['miny'], mandera_boundary.bounds['max
            # Set the longitude range to cover the approximate area of Mandera
            random_longitude = random.uniform(mandera_boundary.bounds['minx'], mandera_boundary.bounds['ma


            if is_coordinate_in_mandera(random_latitude, random_longitude):
                # Found a random coordinate within Mandera, update the dataframe
                mandera.at[index, 'latitude'] = random_latitude
                mandera.at[index, 'longitude'] = random_longitude
                break
```

In [135]:

```
## save dataset
mandera.to_csv('mandera.csv', index=False)
```

4) Turkana

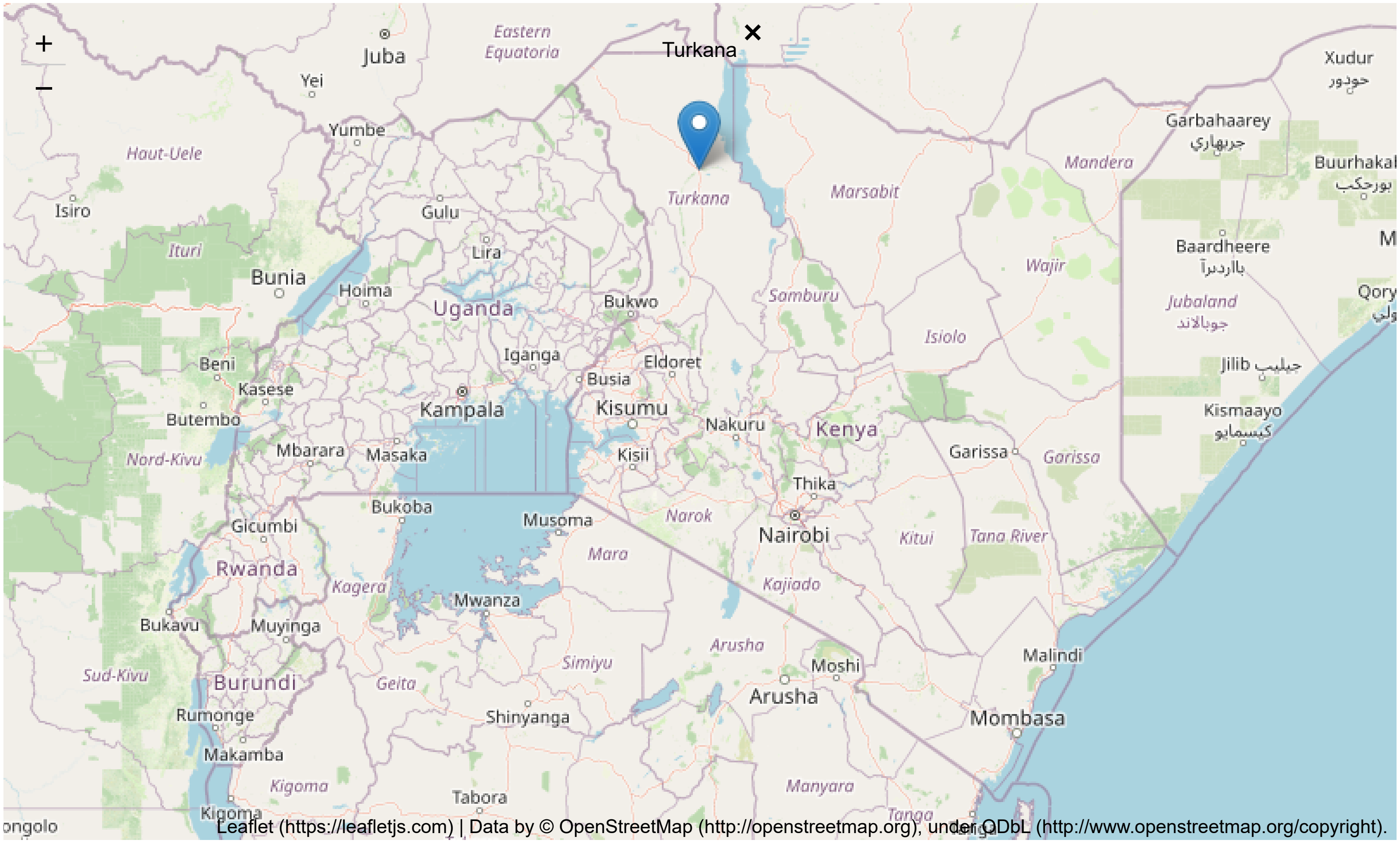
In [136]:

```
# Create a map of Turkana
turkana_map = folium.Map(location=[3.1190, 35.6059], zoom_start=10)

# Add a marker for Turkana town
turkana_marker = folium.Marker(location=[3.1190, 35.6059], popup='Turkana')
turkana_marker.add_to(turkana_map)

# Show the map
turkana_map
```

Out[136]:



In [137]:

```python
# Filter out Marsabit in dataframe
turkana = df[df['county_name'] == "Turkana"]
```

In [138]:

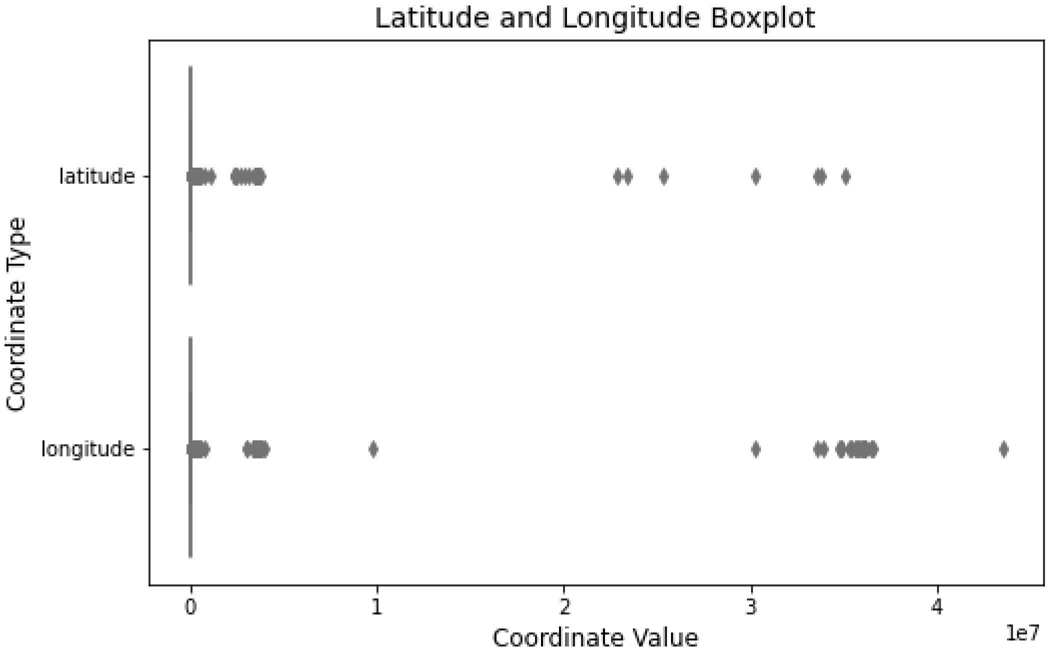
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [139]:

```
# If latitude is greater than longitude they interchange
mask = turkana['latitude'] > turkana['longitude']
turkana.loc[mask, ['latitude', 'longitude']] = turkana.loc[mask, ['longitude', 'latitude']].values
```

In [140]:

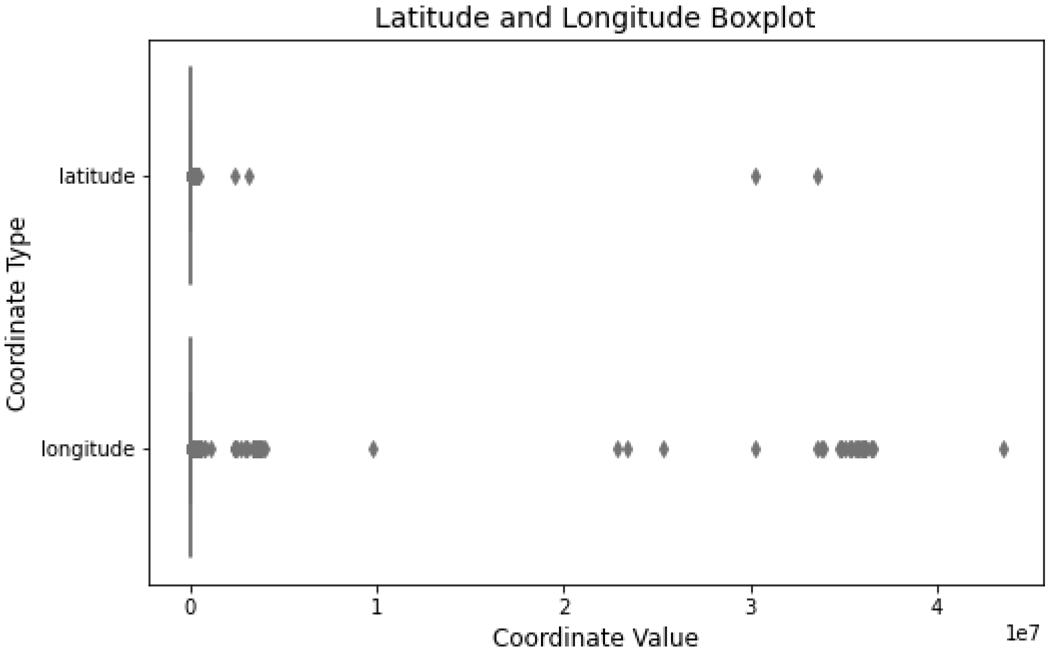
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [141]:

```
# calculate the correction factor for each value
factors = turkana['latitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 1))

# divide each value by its correction factor
turkana['latitude'] = turkana['latitude'] * factors

# # print the corrected dataframe
# print(df_sample)
```

```
<ipython-input-141-2615b8370d35>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  turkana['latitude'] = turkana['latitude'] * factors
```

In [142]:

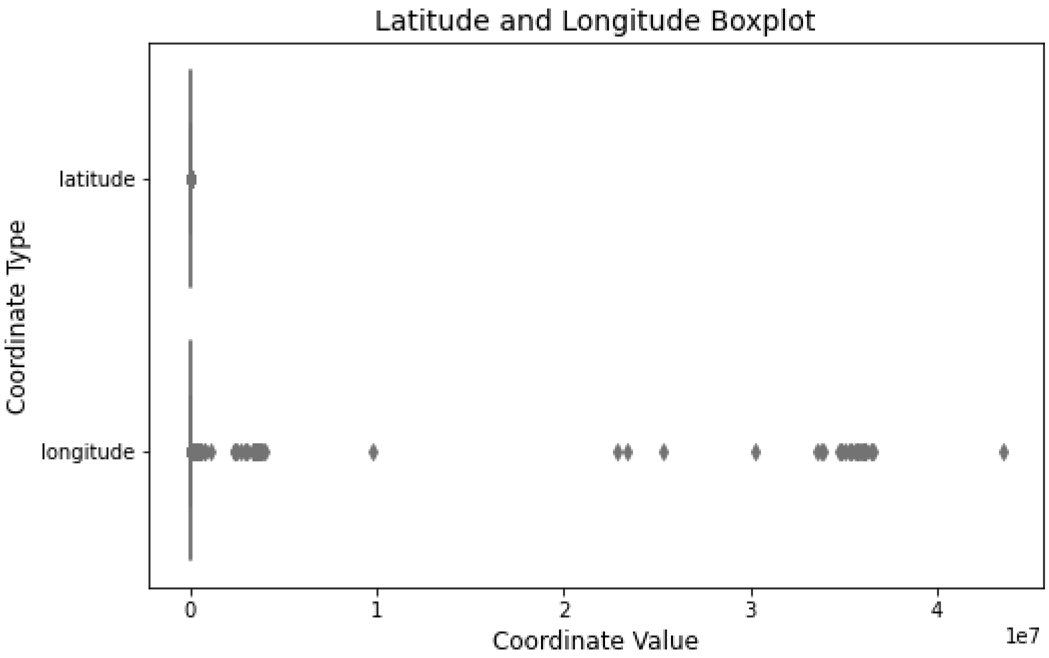
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [143]:

```
# calculate the correction factor for each value
factors = turkana['longitude'].apply(lambda x: 10 ** -(len(str(int(x))) - 2))

# divide each value by its correction factor
turkana['longitude'] = turkana['longitude'] * factors
```

```
<ipython-input-143-e461f9b0d74c>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  turkana['longitude'] = turkana['longitude'] * factors
```

In [144]:

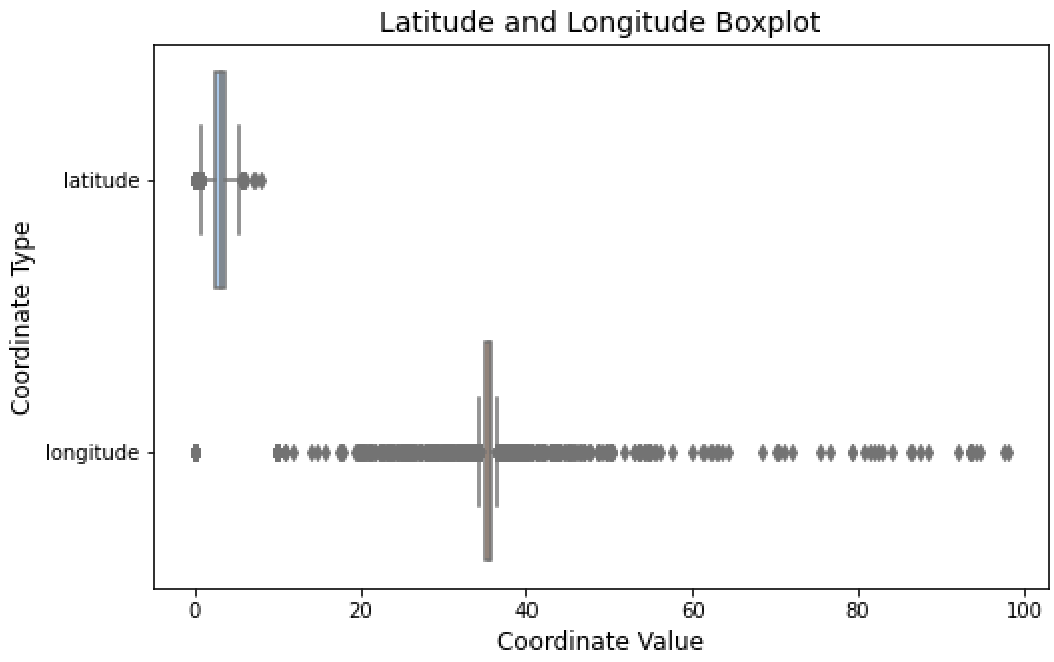
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [145]:

```
# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 10 * turkana['latitude'].std()

# Identify outliers based on the threshold
outliers = ((turkana['latitude'] - mean_latitude).abs() > threshold) | ((turkana['longitude'] - mean_longi

# Replace outliers with random values around the mean
turkana.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=turkana['latitude'].std(), s
turkana.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=turkana['longitude'].std()
```

In [146]:

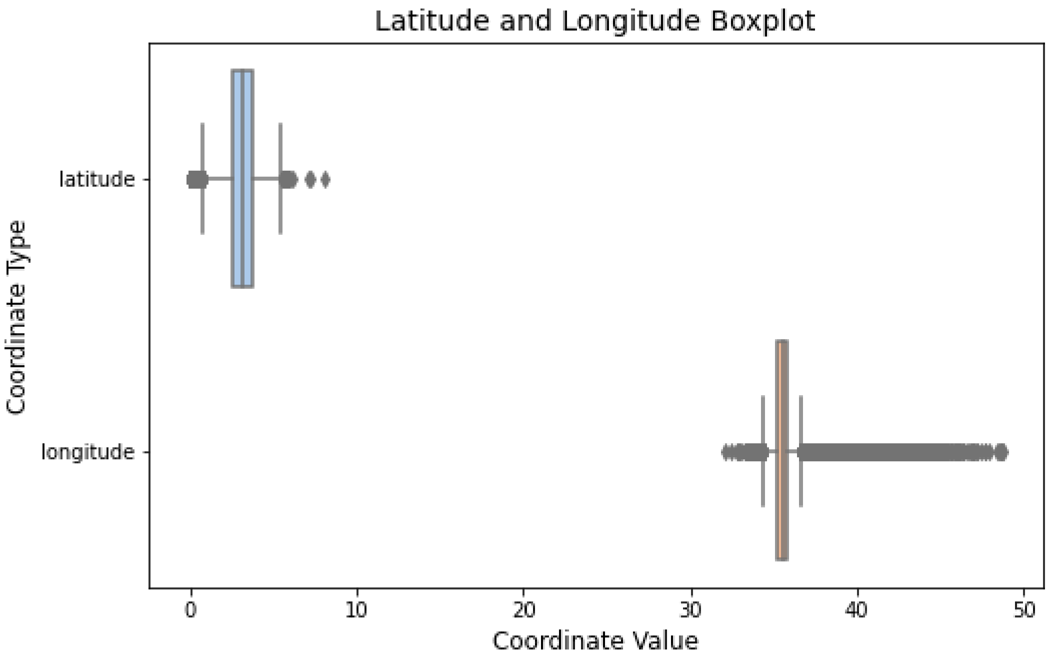
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [147]:

```
# Calculate the mean latitude and longitude
mean_latitude = turkana['latitude'].mean()
mean_longitude = turkana['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * turkana['latitude'].std()

# Identify outliers based on the threshold
outliers = ((turkana['latitude'] - mean_latitude).abs() > threshold) | ((turkana['longitude'] - mean_longi

# Replace outliers with random values around the mean
turkana.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=turkana['latitude'].std(), s
turkana.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=turkana['longitude'].std()
```

In [148]:

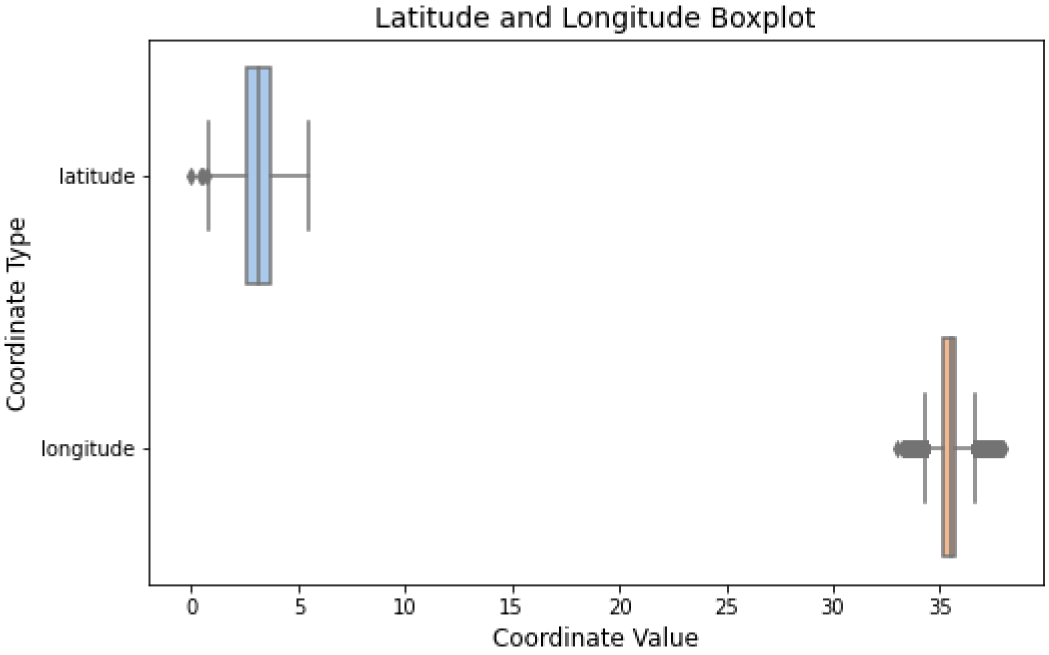
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [149]:

```
# Calculate the mean latitude and longitude
mean_latitude = turkana['latitude'].mean()
mean_longitude = turkana['longitude'].mean()

# Define a threshold for considering outliers (e.g., 3 standard deviations)
threshold = 3 * turkana['latitude'].std()

# Identify outliers based on the threshold
outliers = ((turkana['latitude'] - mean_latitude).abs() > threshold) | ((turkana['longitude'] - mean_longi

# Replace outliers with random values around the mean
turkana.loc[outliers, 'latitude'] = np.random.normal(loc=mean_latitude, scale=turkana['latitude'].std(), s
turkana.loc[outliers, 'longitude'] = np.random.normal(loc=mean_longitude, scale=turkana['longitude'].std()
```

In [150]:

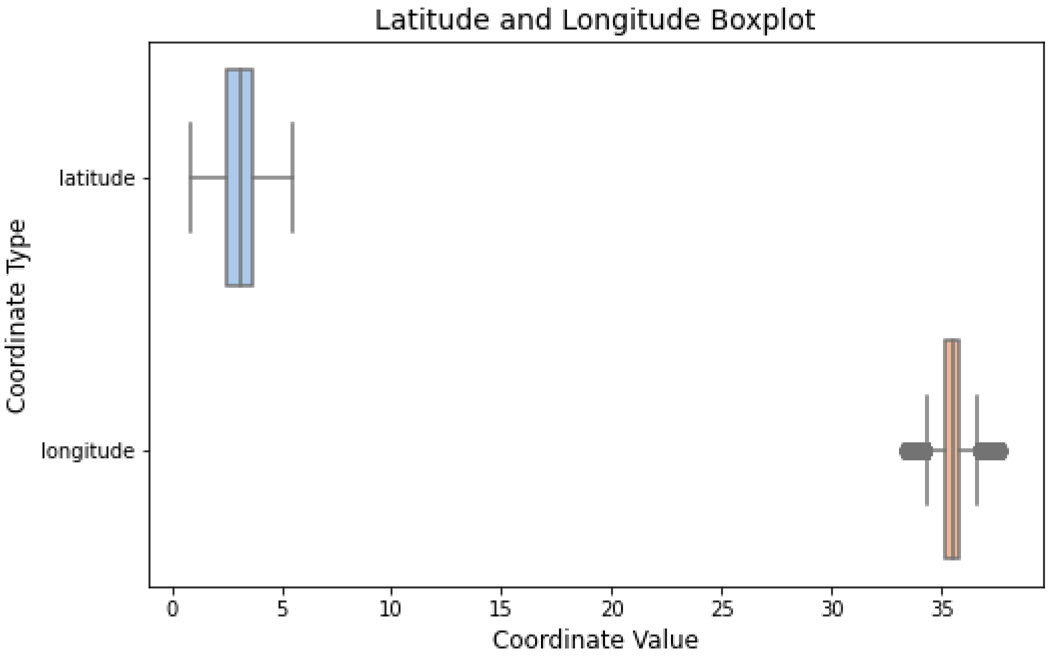
```
# Create a figure with a single subplot
fig, ax = plt.subplots(figsize=(8, 5))

# Create a boxplot for the latitude and longitude columns
sns.boxplot(data=turkana[['latitude', 'longitude']], orient='h', ax=ax, palette='pastel')

# Set the chart title and axis labels
ax.set_title('Latitude and Longitude Boxplot', fontsize=14)
ax.set_xlabel('Coordinate Value', fontsize=12)
ax.set_ylabel('Coordinate Type', fontsize=12)
ax.tick_params(labelsize=10)

# Show the chart
plt.show()
```

```
c:\Users\USER\anaconda3\envs\learn-env\lib\site-packages\seaborn\categorical.py:82: FutureWarning: iteri
tems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```



In [151]:

```
# Read the Turkana boundary shapefile into a GeoDataFrame
turkana_boundary = gpd.read_file('Shapefiles/Turkana_County.shp')  # Replace 'marsabit_boundary.shp' with

# Function to check if a coordinate is in Marsabit
def is_coordinate_in_turkana(latitude, longitude):
    point = Point(longitude, latitude)
    return turkana_boundary.contains(point).any()

# Iterate through the dataframe rows and update coordinates if necessary
for index, row in turkana.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']

    if not is_coordinate_in_turkana(latitude, longitude):
        # Assign random coordinates within Turkana
        while True:
            # Set the latitude range to cover the approximate area of Turkana
            random_latitude = random.uniform(turkana_boundary.bounds['miny'], turkana_boundary.bounds['max
            # Set the longitude range to cover the approximate area of Turkana
            random_longitude = random.uniform(turkana_boundary.bounds['minx'], turkana_boundary.bounds['ma

            if is_coordinate_in_turkana(random_latitude, random_longitude):
                # Found a random coordinate within Turkana, update the dataframe
                turkana.at[index, 'latitude'] = random_latitude
                turkana.at[index, 'longitude'] = random_longitude
                break
```

In [152]:

```
## save dataset
turkana.to_csv('turkana.csv', index=False)
```

In [153]:

```

# Concatenate the dataframes along the row axis
combined_df = pd.concat([mar, mandera, turkana, wajir], ignore_index=True)

# Reset the index of the combined dataframe
combined_df.reset_index(drop=True, inplace=True)
```

In [154]:

```
# Read Excel
combined_df.head()
```

Out[154]:

	household_id	village_id	village_name	sublocation_name	sublocation_id	location_id	location_name	constit
0	4010101010072345100212	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
1	4010101010072345100213	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
2	401010101007234510126	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
3	401010101007234510128	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	
4	401010101007234510133	4010101010072345	Galcha Dida	Nyayo Rd	401010101	4010101	Nagayo	

In [155]:

```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 570675 entries, 0 to 570674
Data columns (total 17 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   household_id       570675 non-null  object
 1   village_id         570675 non-null  int64
 2   village_name       570675 non-null  object
 3   sublocation_name   570675 non-null  object
 4   sublocation_id     570675 non-null  int64
 5   location_id        570675 non-null  int64
 6   location_name      570675 non-null  object
 7   constituency_name  570675 non-null  object
 8   county_name        570675 non-null  object
 9   isbeneficiaryhh    570675 non-null  bool
 10  latitude           570675 non-null  float64
 11  longitude          570675 non-null  float64
 12  ruralurban         544571 non-null  object
 13  constituency_id    570675 non-null  int64
 14  entry_date         570675 non-null  datetime64[ns]
 15  usercode           570675 non-null  object
 16  county_id          570675 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(2), int64(5), object(8)
memory usage: 70.2+ MB
```

In [156]:

```
# Convert village id to object
combined_df['village_id'] = combined_df['village_id'].astype(str)
```

In [157]:

```
# save to excel
combined_df.to_excel('combined_data.xlsx', index=False)
```