Preprocessing/Analysis (28 pts)

Task 1.1 [9 pts] Fill in the missing latitude and longitude values by calculating the average for that country. Round the average to 2 decimal places.

In [58]:
```
import pandas as pd
import csv
from pandas import DataFrame
import math
from collections import OrderedDict
from matplotlib import pyplot as plt

mpgfile = "EuCitiesTemperatures.csv"
#mpgfile = "test.csv"
mpgs = pd.read_csv(mpgfile)
maxCity = {}
maxCityCount = {}

mpgs['latitude'] = mpgs['latitude'].fillna(mpgs.groupby('country')['latitude'
mpgs['longitude'] = mpgs['longitude'].fillna(mpgs.groupby('country')['longitu

mpgs['latitude'] = mpgs['latitude'].round(2)
mpgs['longitude'] = mpgs['longitude'].round(2)
```

Task 1.2 Find out the subset of cities that lie between latitudes 40 to 60 (both inclusive) and longitudes 15 to 30 (both inclusive). Find out which countries have the maximum number of cities in this geographical band. (More than one country could have the maximum number of values.)

In [59]:
```
combined = zip(mpgs2['country'], mpgs2['city'])
mpgs['temperature'] = mpgs['temperature'].fillna(mpgs.groupby(['EU','coastlin

## for rounding
mpgs['temperature'] = mpgs['temperature'].round(2)
cityArray = []
for country,city in combined:
    cityArray.append(city)
    if country not in maxCity.keys():
        maxCity[country] = [city]

    elif country in maxCity.keys():
        maxCity[country].append((city))


print('The subset of cities that lie between latitudes 40 to 60 (both inclusi



mpgs['temperature'] = mpgs['temperature'].fillna(mpgs.groupby(['EU','coastlin
cityArray = []
```

```
## for rounding
mpgs['temperature'] = mpgs['temperature'].round(2)
for country,city in combined:
    cityArray.append(city)
    if country not in maxCity.keys():
        maxCity[country] = [city]

    elif country in maxCity.keys():
        maxCity[country].append((city))


for key in maxCity.keys():
    valueCount = 0
    for value in maxCity[key]:
        valueCount += 1
        maxCityCount[key] = valueCount

mpgs
topCountry = sorted(maxCityCount.items(), key=lambda x: x[1], reverse = True)
greatestValue = topCountry[0][1]
greatestValue = int(greatestValue)

topCountryArray = []
for country, number in topCountry:
    if number == greatestValue:
        topCountryArray.append(country)

print('Countries have the maximum number of cities in this geographical band:



#pd.set_option('display.max_rows', mpgs.shape[0]+1)
#print(mpgs)
```

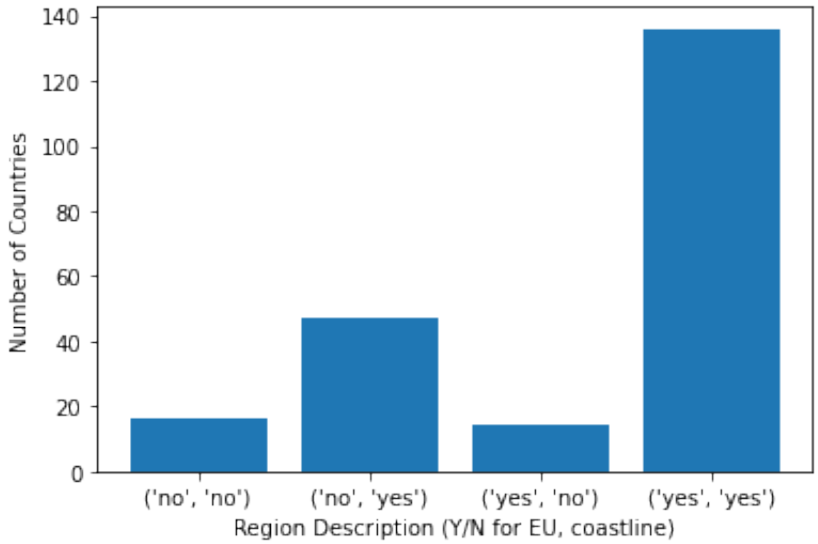
```
The subset of cities that lie between latitudes 40 to 60 (both inclusive) and
longitudes 15 to 30 (both inclusive):
 ['Elbasan', 'Vienna', 'Minsk', 'Orsha', 'Pinsk', 'Brest', 'Hrodna', 'Mazyr',
'Plovdiv', 'Burgas', 'Pleven', 'Ruse', 'Stara Zagora', 'Split', 'Brno', 'Ostra
va', 'Tartu', 'Tallinn', 'Budapest', 'Debrecen', 'Gyor', 'Szeged', 'Bari', 'Fo
ggia', 'Daugavpils', 'Riga', 'Klaipeda', 'Kaunas', 'Vilnius', 'Skopje', 'Balti
', 'Chisinau', 'Podgorica', 'Bialystok', 'Bydgoszcz', 'Bytom', 'Elblag', 'Kiel
ce', 'Koszalin', 'Poznan', 'Wroclaw', 'Warsaw', 'Botosani', 'Braila', 'Buchare
st', 'Craiova', 'Sibiu', 'Arad', 'Bacau', 'Baia Mare', 'Constanta', 'Belgrade'
, 'Nis', 'Novi Sad', 'Bratislava', 'Kosice', 'MalmÃ¶', 'Uppsala', 'Bursa', 'Ed
irne', 'Tekirdag', 'Chernivtsi', 'Rivne']

Countries have the maximum number of cities in this geographical band: ['Polan
d', 'Romania']
```

1.3 Fill in the missing temperature values by the average temperature value of the similar region type. A region type would be a combinaton of whether it is in EU (yes/no) and whether it has a coastline (yes/no).

In [60]:
```
mpgs['temperature'] = mpgs['temperature'].fillna(mpgs.groupby(['EU','coastlin
```

Task 2.1 [6 pts] Plot a bar chart for the number of cities belonging to each of the regions described in Preprocessing/Analysis #3 above.

In [61]:
```
grouped = (mpgs.groupby(['EU','coastline'])).size()
grouped_name = mpgs.groupby(['EU','coastline'])
i=0
size_list = []
key_list = []
for g in grouped:
    size_list.append(g)
for key,item in grouped_name:
    key_list.append(str(key).strip())


plt.bar(key_list, size_list)
plt.xlabel('Region Description (Y/N for EU, coastline)')
plt.ylabel('Number of Countries')
plt.show()
```



1.2.2 [7 pts] Plot a scatter plot of latitude (y-axis) v/s longitude (x-axis) values to get a map-like visual of the cities under consideration. All the cities in the same country should have the same color.

In [63]:
```
## task 1.5
grouped_15 = mpgs.groupby(['country'])

country_dict = {}
long_array = []
lat_array = []

for i in range(len(mpgs)):
    city_15 = mpgs.values[i][0]
```

```
    country_15 = mpgs.values[i][1]
    population_15 = mpgs.values[i][2]
    EU_15 = mpgs.values[i][3]
    coastline_15 = mpgs.values[i][4]
    latitude_15 = mpgs.values[i][5]
    longitude_15 = mpgs.values[i][6]
    temperature_15 = mpgs.values[i][7]
    if country_15 in country_dict.keys():
        country_dict[country_15].append((city_15, longitude_15, latitude_15))
        long_array.append(longitude_15)
        lat_array.append(latitude_15)
    else:
        country_dict[country_15] = []
        country_dict[country_15].append((city_15, longitude_15, latitude_15))
        long_array.append(longitude_15)
        lat_array.append(latitude_15)
```

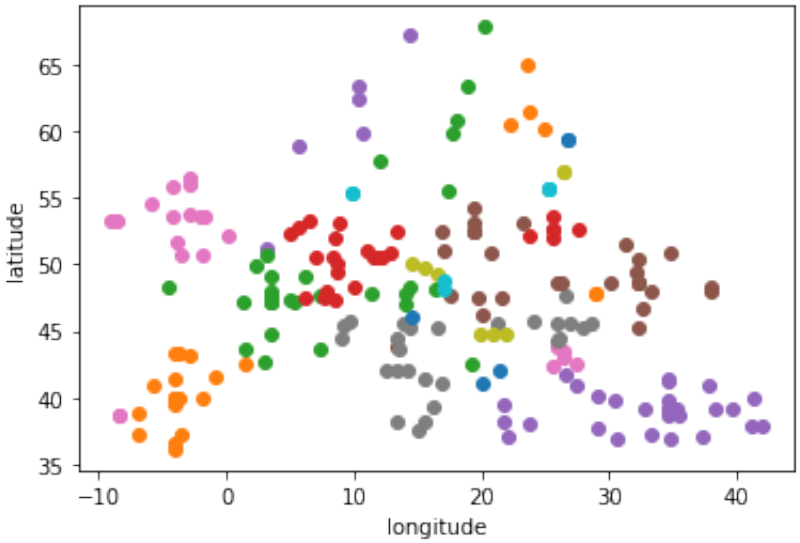
```
#long_15 = mpgs.groupby(['country'])['country','longitude']
```

```
test_dict = {}
#test_dict[1] = np.array((1, 2),(4,5))
#test_dict[2] = np.array((3,3))
```

```
latitude_list= []
long_list = []
latArr = []
longArr = []
```

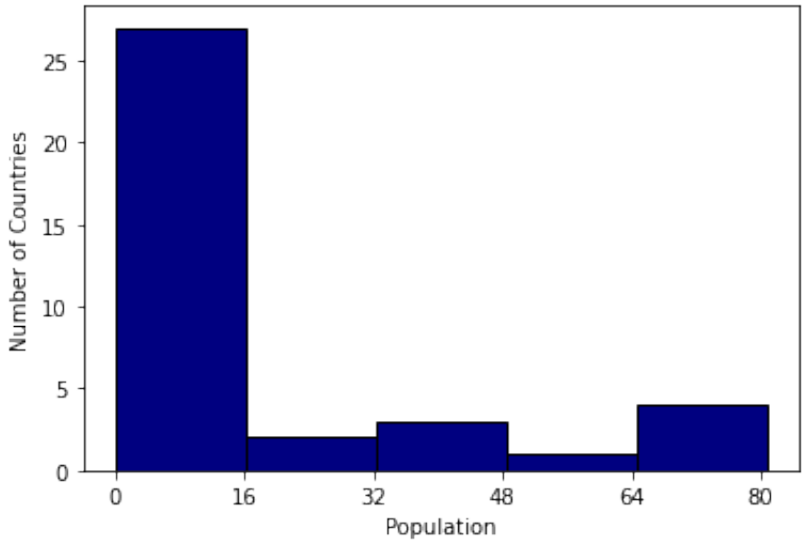
```
for key in country_dict.keys():
    latArr = []
    longArr = []
    for value in country_dict[key]:

        longArr.append(value[1])
        latArr.append(value[2])

    #print(key, latArr, longArr)
    if (len(latArr) > 0) and (len(longArr)> 0):
        plt.scatter(longArr,latArr)
    #latArr = []
    #longArr = []

        #print('long', value[1])
        #print('lat', value[2])
```

```
    #print(split_val[1][:-2], split_val[2][:-2])


    zipped = zip(latitude_list, long_list)
    for a,z in zipped:
        latitude_list.append(z)
        long_list.append(a)

    #plt.scatter(long_list, latitude_list)
    long_list= []
    latitude_list = []


    #for i in len(value):
    #    print('0[]',value[i][0])

   # plt.scatter(country_dict[c][1], country_dict[c][2])
    #plt.scatter()
    #for x_val,y_val,z_val in value:
     #   plt.scatter(value, z_val)



plt.xlabel('longitude')
plt.ylabel('latitude')
plt.show()



## 1.6


#for country,latitude

#print(len(country_dict.keys()))
```



[6 pts] The population column contains values unique to each country. So two cities of the same country will show the same population value. Plot a histogram of the number of countries belonging to each population group: split the population values into 5 bins (groups).

In [54]:
```
## 1.2.3

## The population column contains values unique to each country.
#So two cities of the same country will show the same population value.
#Plot a histogram of the number of countries belonging to each population gro
#split the population values into 5 bins (groups).


population_list = []
for p in mpgs['population']:
    if p not in population_list:
        population_list.append(p)
    else:
        pass


plt.hist(population_list,bins = 5, color="navy", edgecolor="black")
plt.xticks(range(0, int(max(population_list)+1), int(max(population_list)/5))
plt.xlabel('Population')
plt.ylabel('Number of Countries')
plt.show();
```



[8 pts] Plot subplots (2, 2), with proper titles, one each for the region types described in Preprocessing/Analysis #3 above. Each subplot should be a scatter plot of Latitude (y-axis) vs. City (x-axis), where the color of the plot points should be based on the temperature values: 'red' for temperatures above 10, 'blue' for temperatures below 6 and 'orange for temperatures between 6 and 10 (both inclusive). For each subplot, set xticks to an array of numbers from 0 to n-1 (both inclusive), where n is the total number of cities in each region type. This represents each city as a number between 0 and n-1.

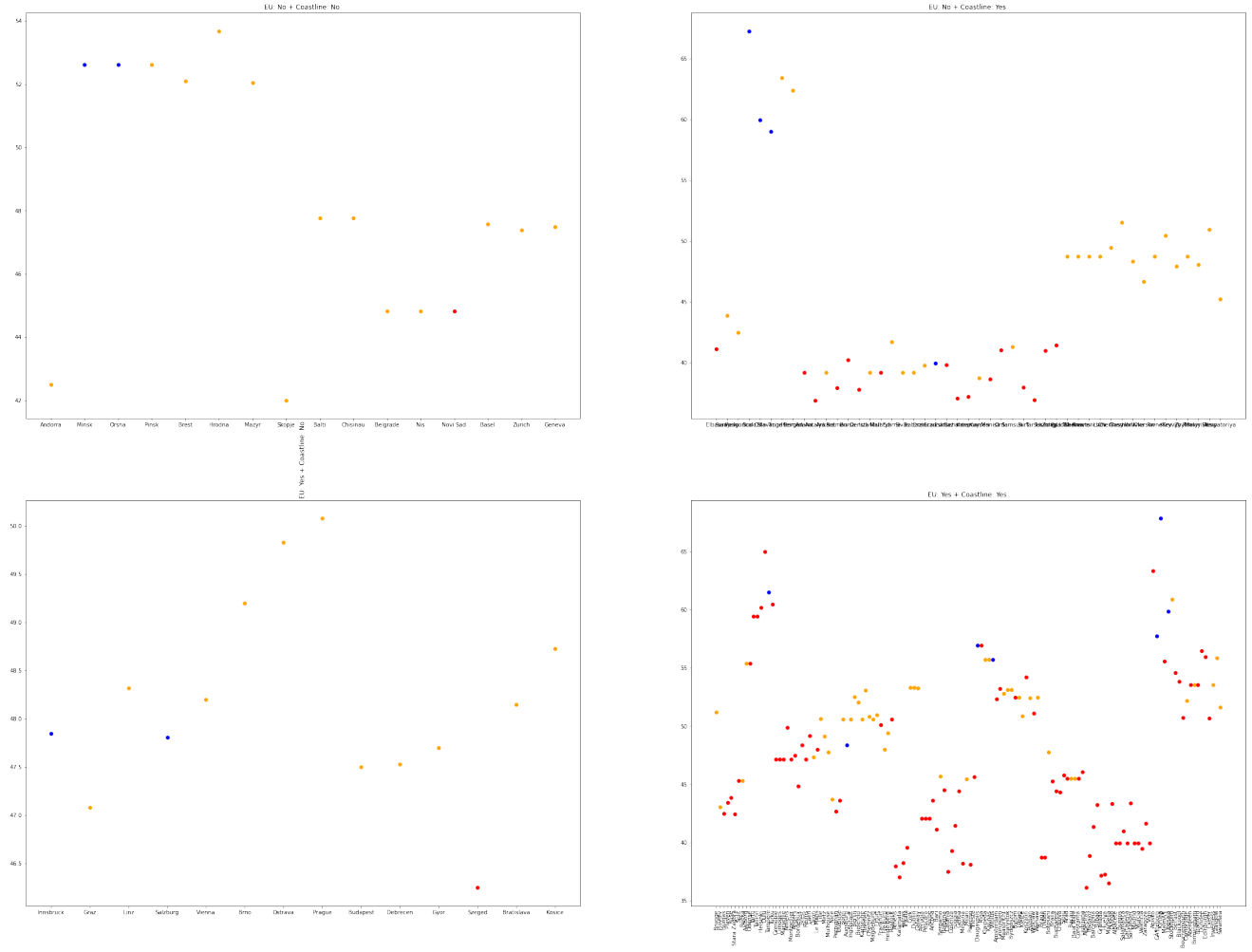
1.2.4

In [56]:
```
#1.2.4
## yes, yes
lat_list_1 = []
city_list_1 = []
temp_list_1 = []

lat_df_1 = mpgs['latitude'].where(mpgs['EU'] == 'yes').where(mpgs['coastline'
city_df_1 = mpgs['city'].where(mpgs['EU'] == 'yes').where(mpgs['coastline'] =
temp_df_1 = mpgs['temperature'].where(mpgs['EU'] == 'yes').where(mpgs['coastl

fig, axes = plt.subplots(2, 2, figsize=(40, 30))

for lat in lat_df_1:
    if type(lat) == float:
        if str(lat)!= 'nan':
            lat_list_1.append(lat)

for city in city_df_1:
    if str(city)!= 'nan':
        city_list_1.append(city)

for temp in temp_df_1:
    if str(temp)!= 'nan':
        temp_list_1.append(temp)
```

```
x = list(range(0,len(city_list_1)))
zipped_temp = zip(x, lat_list_1, temp_list_1)

temp_city_list = []

for x_val,l,t in zipped_temp:

    if t > 10:
        axes[1,1].scatter(x_val,l,color = 'red')
        temp_city_list.append(city_list_1[x_val])
        #plt.xticks(np.arange(0, len(city_list_1)-1), color= 'red')

    if t < 6:
        axes[1,1].scatter(x_val,l,color = 'blue')
        temp_city_list.append(city_list_1[x_val])
        #plt.xticks(np.arange(0, len(city_list_1)-1), color= 'red')

    if 6 <= t<= 10:
        axes[1,1].scatter(x_val,l,color = 'orange')
        temp_city_list.append(city_list_1[x_val])

        #plt.tick_params(axis='x', which='major', labelsize= 3)
        #plt.xticks(rotation=90)

#plt.xticks(x,temp_city_list,rotation = 'vertical', color='orange')
#print(temp_city_list)
axes[1,1].set_xticks(x)
plt.xticks(rotation=90)
axes[1,1].set_xticklabels(temp_city_list, rotation = 90)
#'Oulu', 'Tampere', 'Turku', 'Grenoble', 'Limoges', 'Marseille', 'Amiens', 'M
#plt.margins(0.2)

#print(len(x),len(temp_city_list))

axes[1,1].set_title("EU: Yes + Coastline: Yes ")


## no, yes
lat_list_2 = []
city_list_2 = []
temp_list_2 = []

lat_df_2 = mpgs['latitude'].where(mpgs['EU'] == 'no').where(mpgs['coastline']
city_df_2 = mpgs['city'].where(mpgs['EU'] == 'no').where(mpgs['coastline'] ==
temp_df_2 = mpgs['temperature'].where(mpgs['EU'] == 'no').where(mpgs['coastli


for lat in lat_df_2:
    if type(lat) == float:
        if str(lat)!= 'nan':
            lat_list_2.append(lat)
```

```
for city in city_df_2:
    if str(city)!= 'nan':
        city_list_2.append(city)

for temp in temp_df_2:
    if str(temp)!= 'nan':
        temp_list_2.append(temp)

zipped_temp = zip(city_list_2, lat_list_2, temp_list_2)
for c,l,t in zipped_temp:
    if t > 10:
        axes[0,1].scatter(c,l,color = 'red')

    if t < 6:
        axes[0,1].scatter(c,l,color = 'blue')

    if 6 <= t<= 10:
        axes[0,1].scatter(c,l,color = 'orange')
axes[0,1].set_title("EU: No + Coastline: Yes ")


## yes, no
lat_list_3 = []
city_list_3 = []
temp_list_3 = []

lat_df_3 = mpgs['latitude'].where(mpgs['EU'] == 'yes').where(mpgs['coastline'
city_df_3 = mpgs['city'].where(mpgs['EU'] == 'yes').where(mpgs['coastline'] =
temp_df_3 = mpgs['temperature'].where(mpgs['EU'] == 'yes').where(mpgs['coastl


for lat in lat_df_3:
    if type(lat) == float:
        if str(lat)!= 'nan':
            lat_list_3.append(lat)

for city in city_df_3:
    if str(city)!= 'nan':
        city_list_3.append(city)

for temp in temp_df_3:
    if str(temp)!= 'nan':
        temp_list_3.append(temp)



#axes[1,1].scatter(city_list_1,lat_list_1)
zipped_temp = zip(city_list_3, lat_list_3, temp_list_3)
for c,l,t in zipped_temp:
    if t > 10:
        axes[1,0].scatter(c,l,color = 'red')
```

```
    if t < 6:
        axes[1,0].scatter(c,l,color = 'blue')

    if 6 <= t<= 10:
        axes[1,0].scatter(c,l,color = 'orange')
axes[1,0].set_title("EU: Yes + Coastline: No ", rotation = 90)

## no, no
lat_list_4 = []
city_list_4 = []
temp_list_4 = []

lat_df_4 = mpgs['latitude'].where(mpgs['EU'] == 'no').where(mpgs['coastline']
city_df_4 = mpgs['city'].where(mpgs['EU'] == 'no').where(mpgs['coastline'] ==
temp_df_4 = mpgs['temperature'].where(mpgs['EU'] == 'no').where(mpgs['coastli


for lat in lat_df_4:
    if type(lat) == float:
        if str(lat)!= 'nan':
            lat_list_4.append(lat)

for city in city_df_4:
    if str(city)!= 'nan':
        city_list_4.append(city)

for temp in temp_df_4:
    if str(temp)!= 'nan':
        temp_list_4.append(temp)



#axes[1,1].scatter(city_list_1,lat_list_1)
zipped_temp = zip(city_list_4, lat_list_4, temp_list_4)
for c,l,t in zipped_temp:
    if t > 10:
        axes[0,0].scatter(c,l,color = 'red')

    if t < 6:
        axes[0,0].scatter(c,l,color = 'blue')

    if 6 <= t<= 10:
        axes[0,0].scatter(c,l,color = 'orange')
axes[0,0].set_title("EU: No + Coastline: No ")




plt.show()
```


EU: No + Coastline: No
Andorra
Minsk
Orsha
Pinsk
Brest
Hrodna
Mazyr
Skopje
Salti
Chisinau
Belgrade
Nis
Novi Sad
Basel
Zurich
Geneva
EU: No + Coastline: Yes
EU: Yes + Coastline: No
Innsbruck
Graz
Linz
Salzburg
Vienna
Brno
Ostrava
Prague
Budapest
Debrecen
Gyor
Szeged
Bratislava
Kosice
EU: Yes + Coastline: Yes

In [ ]: