In [1]:

```
#Creating a new Dataframe
import pandas as pd
import numpy as np
import matplotlib.pylab as plt

data = {'year': [2010, 2011, 2012, 2010, 2011, 2012, 2010, 2011, 2012],
 'team': ['FCBarcelona', 'FCBarcelona', 'FCBarcelona', 'RMadrid', 'RMadrid',
'RMadrid', 'ValenciaCF','ValenciaCF', 'ValenciaCF'],
 'wins': [30, 28, 32, 29, 32, 26, 21, 17, 19],
 'draws': [6, 7, 4, 5, 4, 7, 8, 10, 8],
 'losses': [2, 3, 2, 4, 2, 5, 9, 11, 11]
 }
football = pd.DataFrame(
 data, columns=['year', 'team', 'wins', 'draws', 'losses'])
football
```

Out[1]:

	year	team	wins	draws	losses
0	2010	FCBarcelona	30	6	2
1	2011	FCBarcelona	28	7	3
2	2012	FCBarcelona	32	4	2
3	2010	RMadrid	29	5	4
4	2011	RMadrid	32	4	2
5	2012	RMadrid	26	7	5
6	2010	ValenciaCF	21	8	9
7	2011	ValenciaCF	17	10	11
8	2012	ValenciaCF	19	8	11

In [2]:

```
#Question 1: re-create the DataFrame above using the from_dict method.
import pandas as pd
import numpy as np
import matplotlib.pylab as plt

data = {'year': [2010, 2011, 2012, 2010, 2011, 2012, 2010, 2011, 2012],
 'team': ['FCBarcelona', 'FCBarcelona', 'FCBarcelona', 'RMadrid', 'RMadrid',
'RMadrid', 'ValenciaCF','ValenciaCF', 'ValenciaCF'],
 'wins': [30, 28, 32, 29, 32, 26, 21, 17, 19],
 'draws': [6, 7, 4, 5, 4, 7, 8, 10, 8],
 'losses': [2, 3, 2, 4, 2, 5, 9, 11, 11]
 }
football = pd.DataFrame.from_dict(data)
print(football)
```

```
   year         team  wins  draws  losses
0  2010  FCBarcelona    30      6       2
1  2011  FCBarcelona    28      7       3
2  2012  FCBarcelona    32      4       2
3  2010      RMadrid    29      5       4
4  2011      RMadrid    32      4       2
5  2012      RMadrid    26      7       5
6  2010   ValenciaCF    21      8       9
7  2011   ValenciaCF    17     10      11
8  2012   ValenciaCF    19      8      11
```

In [3]:

```
# Reading Tabular Data
edu = pd.read_csv('education_analysis.csv',
 na_values=':', usecols=['TIME', 'GEO', 'Value'])
edu
```

Out[3]:

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
2	2002	European Union (28 countries)	5.00
3	2003	European Union (28 countries)	5.03
4	2004	European Union (28 countries)	4.95
...	...	...	...
379	2007	Finland	5.90
380	2008	Finland	6.10
381	2009	Finland	6.81
382	2010	Finland	6.85
383	2011	Finland	6.76

384 rows × 3 columns

In [4]:

```
#Question 2: read any other file from your hard drive with format other than csv into q_2 DataFrame.
covid = pd.read_excel('Covid.xlsx',
 na_values=':', usecols=['Location', 'Dose Given', 'Full Vac'])
covid
```

Out[4]:

	Location	Dose Given	Full Vac
0	Alabama	6.09M	2.46M
1	Alaska	1.09M	446K
2	Arizona	11.5M	4.37m
3	Arkansas	4.09M	1.61M
4	California	71.5M	27.8M
...	...	...	...
58	60.1% to 65%	25646654	1888245
59	65% to 70%	68013573	2093995
60	70% to 75%	71136310	1014327
61	75% to 80%	32453272	1027695
62	80% to 85%	4880179	673767

63 rows × 3 columns

In [5]:

```
#Viewing Data
edu.head()
```

Out[5]:

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
2	2002	European Union (28 countries)	5.00
3	2003	European Union (28 countries)	5.03
4	2004	European Union (28 countries)	4.95

In [6]:

```
#Viewing Data
edu.tail()
```

Out[6]:

	TIME	GEO	Value
379	2007	Finland	5.90
380	2008	Finland	6.10
381	2009	Finland	6.81
382	2010	Finland	6.85
383	2011	Finland	6.76

In [7]:

```
#Viewing Data
edu.columns
```

Out[7]:

```
Index(['TIME', 'GEO', 'Value'], dtype='object')
```

In [8]:

```
#Viewing Data
edu.index
```

Out[8]:

```
RangeIndex(start=0, stop=384, step=1)
```

In [9]:

```
#Viewing Data
edu.values
```

Out[9]:

```
array([[2000, 'European Union (28 countries)', nan],
       [2001, 'European Union (28 countries)', nan],
       [2002, 'European Union (28 countries)', 5.0],
       ...,
       [2009, 'Finland', 6.81],
       [2010, 'Finland', 6.85],
       [2011, 'Finland', 6.76]], dtype=object)
```

In [10]:

```
#Viewing Data
edu.describe()
```

Out[10]:

	TIME	Value
count	384.000000	361.000000
mean	2005.500000	5.203989

	TIME	Value
std	3.456556	1.021694
min	2000.000000	2.880000
25%	2002.750000	4.620000
50%	2005.500000	5.060000
75%	2008.250000	5.660000

In [11]:

```
#Selection
edu['Value']
```

Out[11]:

```
0        NaN
1        NaN
2       5.00
3       5.03
4       4.95
        ... 
379     5.90
380     6.10
381     6.81
382     6.85
383     6.76
Name: Value, Length: 384, dtype: float64
```

In [12]:

```
#Selection
edu[10:14]
```

Out[12]:

	TIME	GEO	Value
10	2010	European Union (28 countries)	5.41
11	2011	European Union (28 countries)	5.25
12	2000	European Union (27 countries)	4.91
13	2001	European Union (27 countries)	4.99

In [13]:

```
#Selection
edu.iloc[90:94,:]
```

Out[13]:

	TIME	GEO	Value
90	2006	Belgium	5.98
91	2007	Belgium	6.00
92	2008	Belgium	6.43
93	2009	Belgium	6.57

In [14]:

```
#Filtering Data
edu[edu['Value'] > 6.5].tail()
```

Out[14]:

	TIME	GEO	Value
286	2010	Malta	6.74
287	2011	Malta	7.96
381	2009	Finland	6.81
382	2010	Finland	6.85
383	2011	Finland	6.76

In [15]:

```
#Filtering Missing Values
edu[edu['Value'].isnull()].head()
```

Out[15]:

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
36	2000	Euro area (18 countries)	NaN
37	2001	Euro area (18 countries)	NaN
48	2000	Euro area (17 countries)	NaN

In [16]:

```
#Manipulating Data
edu.max(axis=0)
```

Out[16]:

```
TIME       2011
GEO       Spain
Value      8.81
dtype: object
```

In [17]:

```
#Question 3: Calculate the average and standard deviation values of column ‘Value’ in
#edu DataFrame.
print('Average:', edu['Value'].mean())
print('Standard Deviation:', edu['Value'].std())
```

```
Average: 5.203988919667592
Standard Deviation: 1.0216944748195942
```

In [18]:

```
#Manipulating Data
print('Pandas max function:', edu['Value'].max())
print('Python max function:', max(edu['Value']))
```

```
Pandas max function: 8.81
Python max function: nan
```

In [19]:

```
#Manipulating Data
s = edu['Value'] / 100
s.head()
```

Out[19]:

```
0       NaN
1       NaN
2    0.0500
3    0.0503
4    0.0495
Name: Value, dtype: float64
```

In [20]:

```
#Manipulating Data
s = edu['Value'].apply(np.sqrt)
s.head()
```

Out[20]:

```
0         NaN
1         NaN
2    2.236068
3    2.242766
4    2.224860
Name: Value, dtype: float64
```

In [21]:

```
#Question 4: Calculate the ceil of the scalar sqrt(Value) in the previous example
edu['sq'] = edu['Value'].apply(np.sqrt)
ceil= edu['sq'].apply(np.ceil)
print('ceil of the scalar sqrt(Value):\n',ceil)
```

```
ceil of the scalar sqrt(Value):
 0       NaN
1       NaN
2       3.0
3       3.0
4       3.0
       ...
379     3.0
380     3.0
381     3.0
382     3.0
383     3.0
```

In [22]:

```
#Manipulating Data
s = edu['Value'].apply(lambda d: d**2)
s.head()
```

Out[22]:

```
0        NaN
1        NaN
2    25.0000
3    25.3009
4    24.5025
Name: Value, dtype: float64
```

In [23]:

```
#Manipulating Data
edu['ValueNorm'] = edu['Value'] / edu['Value'].max()
edu.tail()
```

Out[23]:

	TIME	GEO	Value	sq	ValueNorm
379	2007	Finland	5.90	2.428992	0.669694
380	2008	Finland	6.10	2.469818	0.692395
381	2009	Finland	6.81	2.609598	0.772985
382	2010	Finland	6.85	2.617250	0.777526
383	2011	Finland	6.76	2.600000	0.767310

In [24]:
```
#Manipulating Data
edu.drop('ValueNorm', axis=1, inplace=True)
edu.head()
```

Out[24]:

	TIME	GEO	Value	sq
0	2000	European Union (28 countries)	NaN	NaN
1	2001	European Union (28 countries)	NaN	NaN
2	2002	European Union (28 countries)	5.00	2.236068
3	2003	European Union (28 countries)	5.03	2.242766
4	2004	European Union (28 countries)	4.95	2.224860

In [25]:
```
#Manipulating Data
edu = edu.append({'TIME': 2000, 'Value': 5.00, 'GEO': 'a'}, ignore_index=True)
edu.tail()
```

Out[25]:

	TIME	GEO	Value	sq
380	2008	Finland	6.10	2.469818
381	2009	Finland	6.81	2.609598
382	2010	Finland	6.85	2.617250
383	2011	Finland	6.76	2.600000
384	2000	a	5.00	NaN

In [26]:
```
#Manipulating Data
eduFillNa = edu.fillna(0)
eduFillNa .head(10)
```

Out[26]:

	TIME	GEO	Value	sq
0	2000	European Union (28 countries)	0.00	0.000000
1	2001	European Union (28 countries)	0.00	0.000000
2	2002	European Union (28 countries)	5.00	2.236068
3	2003	European Union (28 countries)	5.03	2.242766

	TIME	GEO	Value	sq
4	2004	European Union (28 countries)	4.95	2.224860
5	2005	European Union (28 countries)	4.92	2.218107
6	2006	European Union (28 countries)	4.91	2.215852
7	2007	European Union (28 countries)	4.92	2.218107
8	2008	European Union (28 countries)	5.04	2.244994

In [27]:

```python
#Manipulating Data
edu.drop(max(edu.index), axis=0, inplace=True)
edu.tail()
```

Out[27]:

	TIME	GEO	Value	sq
379	2007	Finland	5.90	2.428992
380	2008	Finland	6.10	2.469818
381	2009	Finland	6.81	2.609598
382	2010	Finland	6.85	2.617250
383	2011	Finland	6.76	2.600000

In [28]:

```python
#Manipulating Data
eduDrop = edu.dropna(how='any', subset=['Value'], axis=0)
eduDrop.head()
```

Out[28]:

	TIME	GEO	Value	sq
2	2002	European Union (28 countries)	5.00	2.236068
3	2003	European Union (28 countries)	5.03	2.242766
4	2004	European Union (28 countries)	4.95	2.224860
5	2005	European Union (28 countries)	4.92	2.218107
6	2006	European Union (28 countries)	4.91	2.215852

In [29]:

```
#Sorting
edu.sort_values(by='Value', ascending=False, inplace=True)
edu.head()
```

Out[29]:

	TIME	GEO	Value	sq
130	2010	Denmark	8.81	2.968164
131	2011	Denmark	8.75	2.958040
129	2009	Denmark	8.74	2.956349
121	2001	Denmark	8.44	2.905168
122	2002	Denmark	8.44	2.905168

In [30]:

```
# Return in original order
edu.sort_index(axis=0, ascending=True, inplace=True)
edu.head()
```

Out[30]:

	TIME	GEO	Value	sq
0	2000	European Union (28 countries)	NaN	NaN
1	2001	European Union (28 countries)	NaN	NaN
2	2002	European Union (28 countries)	5.00	2.236068
3	2003	European Union (28 countries)	5.03	2.242766
4	2004	European Union (28 countries)	4.95	2.224860

In [31]:

```
#Grouping Data
group = edu[['GEO', 'Value']].groupby('GEO').mean()
group.head()
```

Out[31]:

	Value
GEO	
Austria	5.618333
Belgium	6.189091
Bulgaria	4.093333

	Value
GEO	

In [32]:

```
#Question 5: Calculate the mean of the values per year
group = edu[['TIME', 'Value']].groupby('TIME').mean()
group
```

Out[32]:

	Value
TIME	
2000	4.917917
2001	5.085172
2002	5.125625
2003	5.142812
2004	5.067500
2005	5.091563
2006	5.129000
2007	4.998387
2008	5.266897
2009	5.603667
2010	5.549333
2011	5.443667

In [33]:

```
#Rearranging Data
filtered_data = edu[edu['TIME'] > 2005]
pivedu = pd.pivot_table(filtered_data, values='Value',
 index=['GEO'], columns=['TIME'])
pivedu.head()
```

Out[33]:

TIME	2006	2007	2008	2009	2010	2011
GEO						
Austria	5.40	5.33	5.47	5.98	5.91	5.80

TIME	2006	2007	2008	2009	2010	2011
GEO						
Belgium	5.98	6.00	6.43	6.57	6.58	6.55

In [34]:

```python
#using new index to select specific rows by label, using the loc operator:
pivedu.loc[['Spain', 'Portugal'], [2006, 2011]]
```

Out[34]:

TIME	2006	2011
GEO		
Spain	4.26	4.82
Portugal	5.07	5.27

In [35]:

```python
#Ranking Data
pivedu = pivedu.drop(['Euro area (13 countries)',
 'Euro area (15 countries)',
 'Euro area (17 countries)',
 'Euro area (18 countries)',
 'European Union (25 countries)',
 'European Union (27 countries)',
'European Union (28 countries)'
 ], axis=0)
pivedu = pivedu.rename(
 index={'Germany (until 1990 former territory of the FRG)': 'Germany'})
pivedu = pivedu.dropna()
pivedu.rank(ascending=False, method='first').head()
```

Out[35]:

TIME	2006	2007	2008	2009	2010	2011
GEO						
Austria	10.0	7.0	11.0	7.0	8.0	8.0
Belgium	5.0	4.0	3.0	4.0	5.0	5.0
Bulgaria	21.0	21.0	20.0	20.0	22.0	22.0
Cyprus	2.0	2.0	2.0	2.0	2.0	3.0
Czech Republic	19.0	20.0	21.0	21.0	20.0	19.0

In [36]:

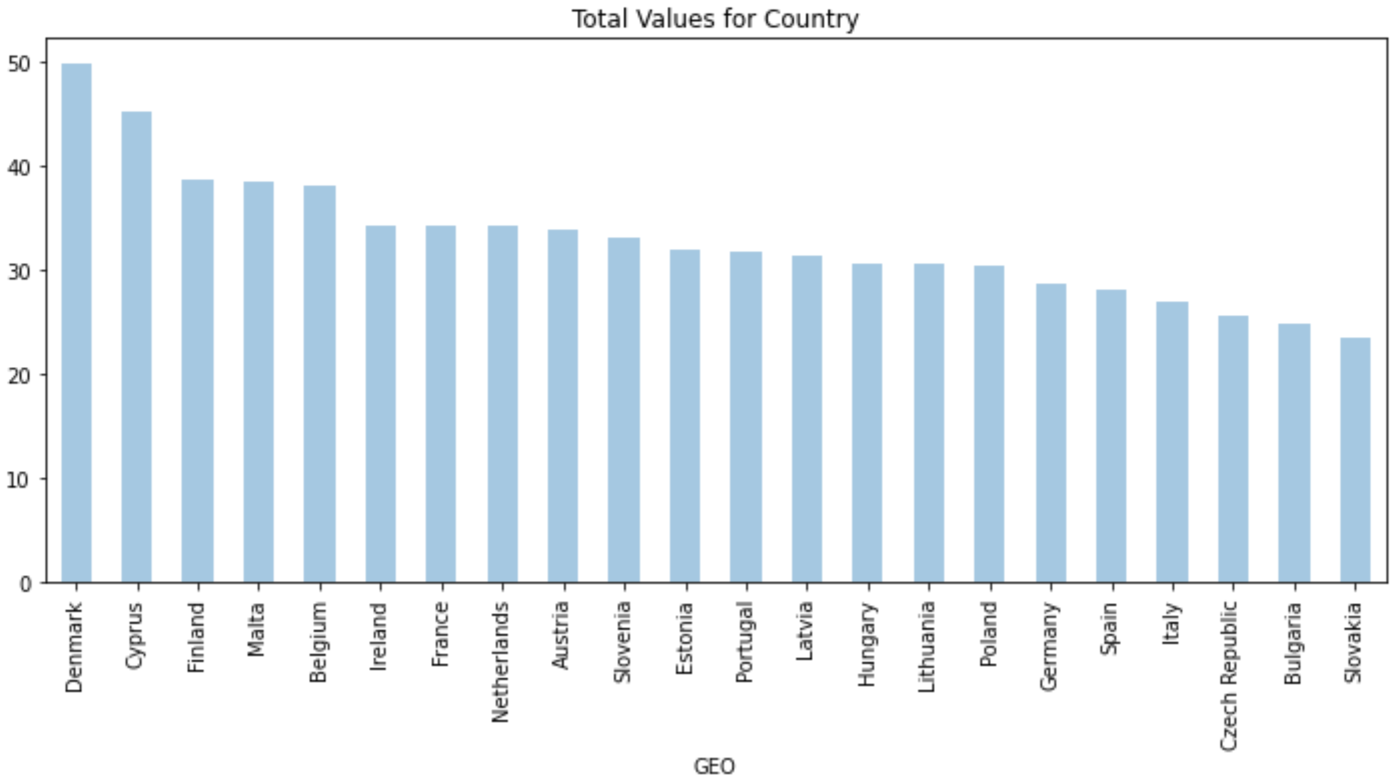
```
#Sumup all the columns and rank the result
totalSum = pivedu.sum(axis=1)
totalSum.rank(ascending=False, method='dense').sort_values().head()
```

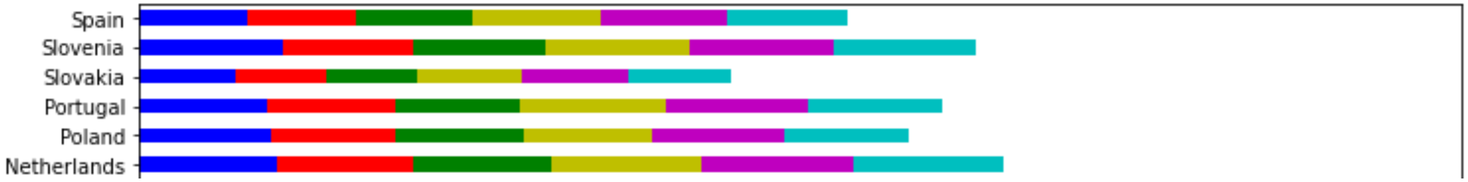
Out[36]:

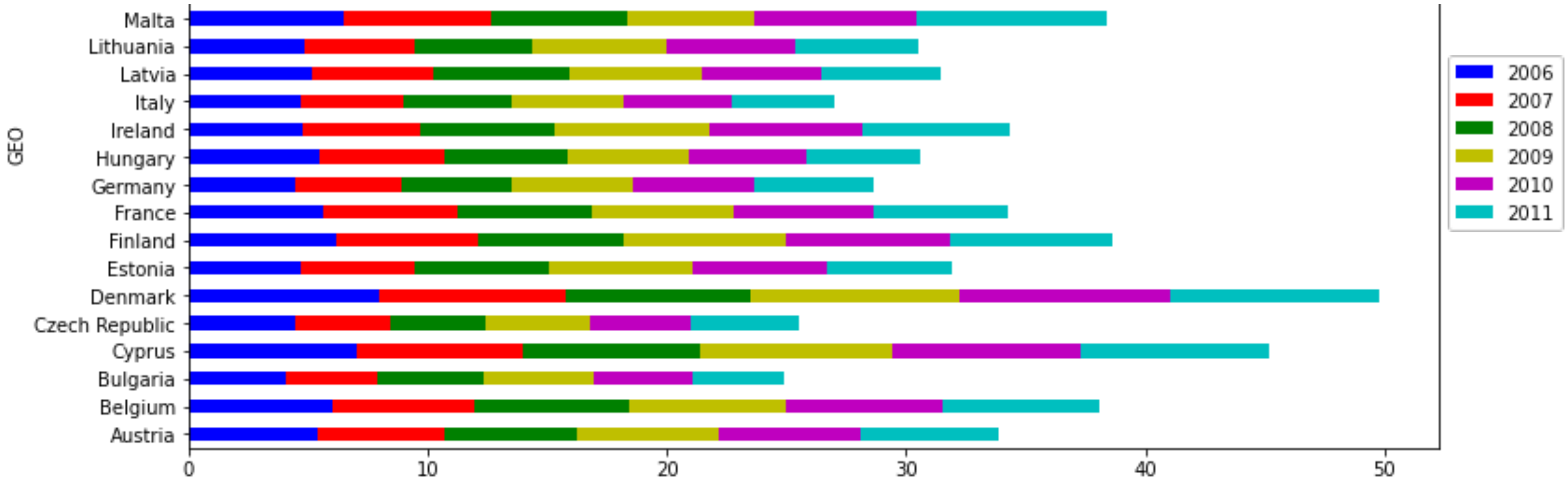
```
GEO
Denmark     1.0
Cyprus      2.0
Finland     3.0
Malta       4.0
Belgium     5.0
dtype: float64
```

```
In [37]: #Plotting
         totalSum = pivedu.sum(axis=1)
         totalSum.rank(ascending=False, method='dense').sort_values().head()
         fig = plt.figure(figsize=(12, 5))
         totalSum = pivedu.sum(axis=1).sort_values(ascending=False)
         totalSum.plot(kind='bar', style='b', alpha=0.4,title='Total Values for Country')
         plt.savefig('Totalvalue_Country.png', dpi=300, bbox_inches='tight')
         my_colors = ['b', 'r', 'g', 'y', 'm', 'c']
         ax = pivedu.plot(kind='barh', stacked=True, color=my_colors, figsize=(12, 6))
         ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

```
Out[37]: <matplotlib.legend.Legend at 0x2796204b400>
```






GEO
Malta
Lithuania
Latvia
Italy
Ireland
Hungary
Germany
France
Finland
Estonia
Denmark
Czech Republic
Cyprus
Bulgaria
Belgium
Austria
0
10
20
30
40
50
2006
2007
2008
2009
2010
2011