In [1]:

```
import numpy as np
import pandas as pd
import ast
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
from itertools import combinations
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from sklearn import linear_model as lm
from scipy.misc import factorial
import matplotlib
%matplotlib notebook
```

Load Global Terrorism Database an filter colums

In [2]:

```
df = pd.read_csv('globalTerrorism.csv', engine = 'python')
dataset1 = pd.concat([df['country'],df['country_txt'],df['region'],df['provstate
'],df['city'],df['targtype1'],df['attacktype1'],df['nkill'],df['nwound']] ,axis=
1)
happiness = pd.read_csv('World-Happiness-Report/2017.csv', engine = 'python')
```

In [3]:

```
def plotHistDist(func, x, r, title, l, xlabel, ylabel):
    plt.hist(r, normed=True, histtype='stepfilled', alpha=0.2)
    plotDist(x, func, title , l, xlabel, ylabel)
#given functions
def plotDist(x, func , title , l , xlabel , ylabel):
    plt.plot(x, func, 'b', lw=2, alpha=0.6, label=l)
    xl = plt.gca().get_xlim()
    #lines on Y-axis
    plt.hlines(0, xl[0], xl[1], linestyles='--', colors='#999999')
    plt.gca().set_xlim ( xl )
    plt.legend(loc='best', frameon=False)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
```

In [4]:

```
#Making dataFrame of counts of attacks in each country
terrCnt = dataset1.country_txt.value_counts()
dfCnt = pd.DataFrame(terrCnt)
dfCnt = dfCnt.rename(columns={'country_txt':'NumberofAttacks'})
```

US probability of attacks

In [5]:

```
#Poisson distribution
tot = 0
arr = [0,0,0,0,0,0,0,0,0]
for i in range(9):
    UnitedStates = df.loc[df['country'] == 217]
    GT1970 = UnitedStates[UnitedStates['iyear'] > 1972+(5*i)]
    fcnt = GT1970[GT1970['iyear'] < 1977+(5*i)]
    fcnt = fcnt.country_txt.value_counts()
    arr[i] = fcnt[0]
    tot += fcnt[0]

avg = tot/9
print("The probability there will be exactly 200 attacks in the US within the ne
xt five-year period:")
print(stats.poisson.pmf(200, avg))


total = 0
for i in range(200):
    tmp = stats.poisson.pmf(i,avg)
    total += tmp
print("\nThe probability there will be more than 200 attacks in the US within th
e next five-year period:")
probability = 1 - total
print(probability)
```

```
The probability there will be exactly 200 attacks in the US within t
he next five-year period:
0.008836103705157198

The probability there will be more than 200 attacks in the US within
the next five-year period:
0.06685510229484393
```

Spain probability of attacks

In [6]:

```
#Poisson distribution
tot = 0
for i in range(9):
    UnitedStates = df.loc[df['country'] == 185]
    GT1970 = UnitedStates[UnitedStates['iyear'] > 1972+(5*i)]
    fcnt = GT1970[GT1970['iyear'] < 1977+(5*i)]
    fcnt = fcnt.country_txt.value_counts()
    tot += fcnt[0]
    arr[i] = fcnt[0]

avg = tot/9
arr = np.arange(3000)

print("The probability there will be exactly 200 attacks in Spain within the nex
t five-year period:")
print(stats.poisson.pmf(200, avg))


total = 0
for i in range(250):
    tmp = stats.poisson.pmf(i,avg)
    total += tmp
print("\nThe probability there will be more than 250 attacks in Spain within the
next five-year period:")
print(1 - total)
```

```
The probability there will be exactly 200 attacks in Spain within th
e next five-year period:
8.031526999655393e-09

The probability there will be more than 250 attacks in Spain within
the next five-year period:
0.9896156867013342
```

Iran probability of attacks

In [7]:

```
#Poisson distribution
tot = 0
for i in range(9):
    UnitedStates = df.loc[df['country'] == 94]
    GT1970 = UnitedStates[UnitedStates['iyear'] > 1972+(5*i)]
    fcnt = GT1970[GT1970['iyear'] < 1977+(5*i)]
    fcnt = fcnt.country_txt.value_counts()
    tot += fcnt[0]

avg = tot/9
arr = np.arange(3000)

print("The probability there will be exactly 90 attacks in Iran within the next five-year period:")
print(stats.poisson.pmf(90, avg))


total = 0
for i in range(50):
    tmp = stats.poisson.pmf(i,avg)
    total += tmp
print("\nThe probability there will be more than 50 attacks in Iran within the next five-year period:")
print(1 - total)
```

```
The probability there will be exactly 90 attacks in Iran within the
next five-year period:
0.00046439888630927113

The probability there will be more than 50 attacks in Iran within th
e next five-year period:
0.9725611492330197
```

Iraq probability of attacks

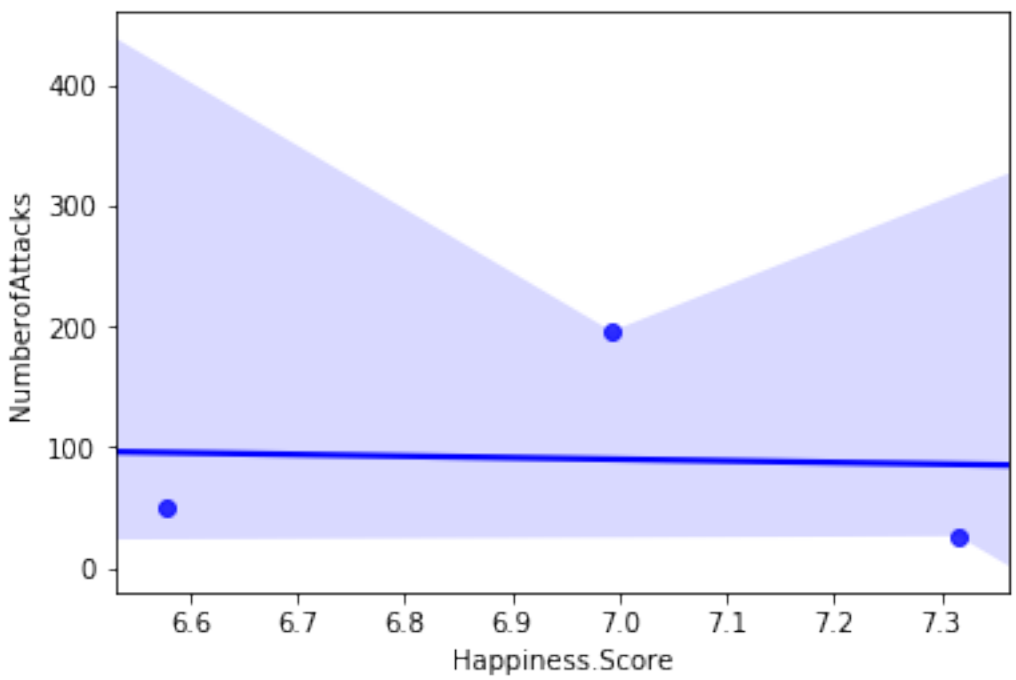
In [8]:

```
#Poisson distribution
tot = 0
arr = [0,0,0,0,0,0,0,0,0]
for i in range(9):
    UnitedStates = df.loc[df['country'] == 95]
    GT1970 = UnitedStates[UnitedStates['iyear'] > 1972+(5*i)]
    fcnt = GT1970[GT1970['iyear'] < 1977+(5*i)]
    #print(GT1970)
    fcnt = fcnt.country_txt.value_counts()
    arr[i] = fcnt[0]
    tot += fcnt[0]

avg = tot/9
print("The probability there will be exactly 2100 attacks in Iraq within the nex
t five-year period:")
print(stats.poisson.pmf(2100, avg))


total = 0
for i in range(2000):
    tmp = stats.poisson.pmf(i,avg)
    total += tmp
print("\nThe probability there will be more than 2000 attacks in Iraq within the
next five-year period:")
print(1 - total)
```

```
The probability there will be exactly 2100 attacks in Iraq within th
e next five-year period:
0.002001735073177817

The probability there will be more than 2000 attacks in Iraq within
the next five-year period:
0.99995349841246
```

In [9]:

```
#NorthAmerica Happiness score vs number of attacks
No = df.loc[df['iyear'] > 2013]
NorthAmerica = No.loc[No['region'] == 1]
NorthAmericaTerrCnt = NorthAmerica.country_txt.value_counts()
dfNA = pd.DataFrame(NorthAmericaTerrCnt)
dfNA = dfNA.rename(columns={'country_txt':'NumberofAttacks'})
dfNA
happiness = pd.read_csv('World-Happiness-Report/2017.csv', engine = 'python')
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfNA],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})
combined
slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
graph = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, col
or='b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})
plt.show()
print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```




```
Slope is -13
Intercept is 182
R value is -0.053140
P value is 0.966154
Std Error is 248
```

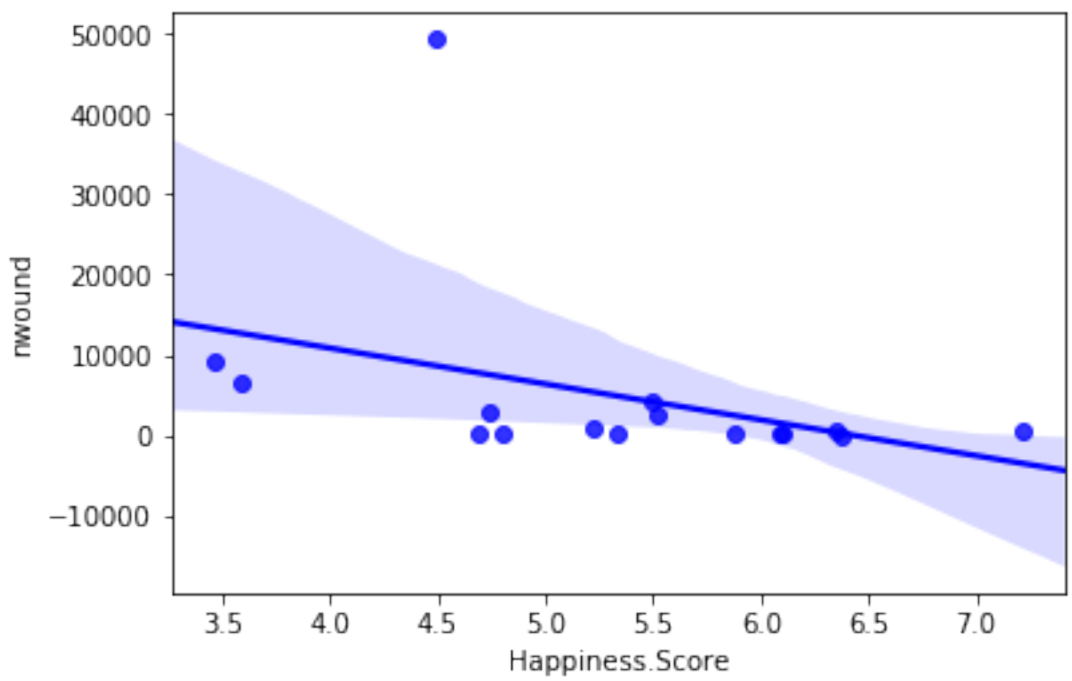
Happiness Score vs Wounded

In [10]:

```
#Happiness score vs wounded
df = df.loc[df['iyear'] > 2013]
wound = df.loc[df['region'] == 10]
wound = pd.concat([wound['country_txt'],wound['nwound']],axis=1)
filtered = wound.loc[df['nwound'] > 0]
pwound = filtered.set_index('country_txt')
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
#cant use df because its a different dataset! use Happiness dataset
reducedPwound = pwound.groupby(pwound.index).sum()
wounded = pd.concat([phappiness,reducedPwound],axis=1,join='inner')
print(wounded)
slope, intercept, r_value, p_value, std_err = stats.linregress(wounded['Happines
s.Score'],wounded['nwound'])
ax = sns.regplot(x="Happiness.Score", y="nwound", data=wounded, color='b', line_
kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})
#wounded
print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

	Happiness.Score	nwound
Israel	7.213	460.0
Qatar	6.375	1.0
Saudi Arabia	6.344	569.0
Kuwait	6.105	228.0
Bahrain	6.087	89.0
Algeria	5.872	70.0
Libya	5.525	2651.0
Turkey	5.500	4292.0
Jordan	5.336	84.0
Lebanon	5.225	1002.0
Tunisia	4.805	216.0
Egypt	4.735	2854.0
Iran	4.692	91.0
Iraq	4.497	49334.0
Yemen	3.593	6433.0
Syria	3.462	9277.0

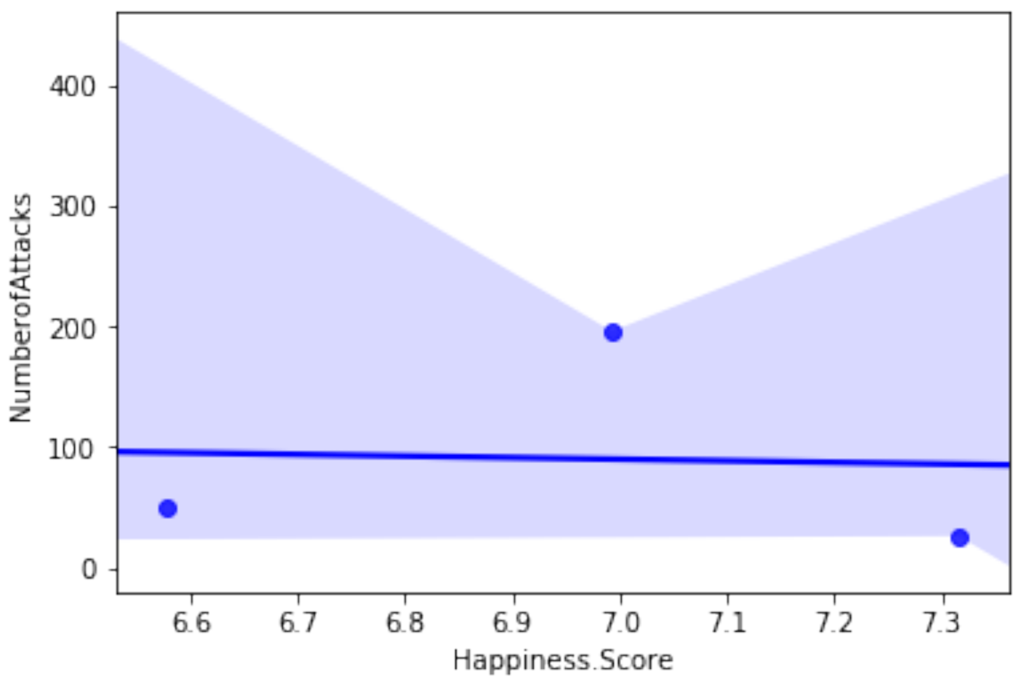
```
Slope is -4467
Intercept is 28690
R value is -0.372715
P value is 0.155094
Std Error is 2972
```



NorthAmerica Happiness score vs number of attacks

In [11]:

```
No = df.loc[df['iyear'] > 2013]
NorthAmerica = No.loc[No['region'] == 1]
NorthAmericaTerrCnt = NorthAmerica.country_txt.value_counts()
dfNA = pd.DataFrame(NorthAmericaTerrCnt)
dfNA = dfNA.rename(columns={'country_txt':'NumberofAttacks'})
dfNA
happiness = pd.read_csv('World-Happiness-Report/2017.csv', engine = 'python')
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfNA],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})
combined

slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
graph = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, col
or='b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})
plt.show()

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```



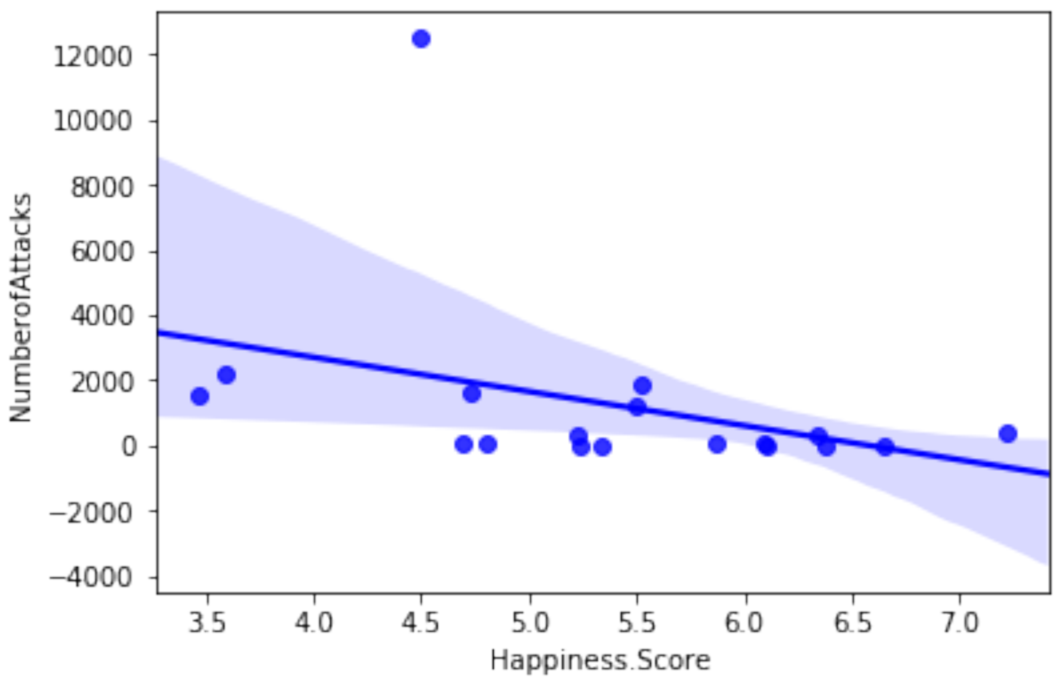

```
Slope is -13
Intercept is 182
R value is -0.053140
P value is 0.966154
Std Error is 248
```

Middle East Happiness score vs number of attacks

In [12]:

```
MiddleEast = df.loc[df['region'] == 10]
MiddleEastTerrCnt = MiddleEast.country_txt.value_counts()
dfME = pd.DataFrame(MiddleEastTerrCnt)
dfME = dfME.rename(columns={'country_txt':'NumberofAttacks'})
dfME
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfME],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})

slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is -1049
Intercept is 6905
R value is -0.361460
P value is 0.140529
Std Error is 676
```

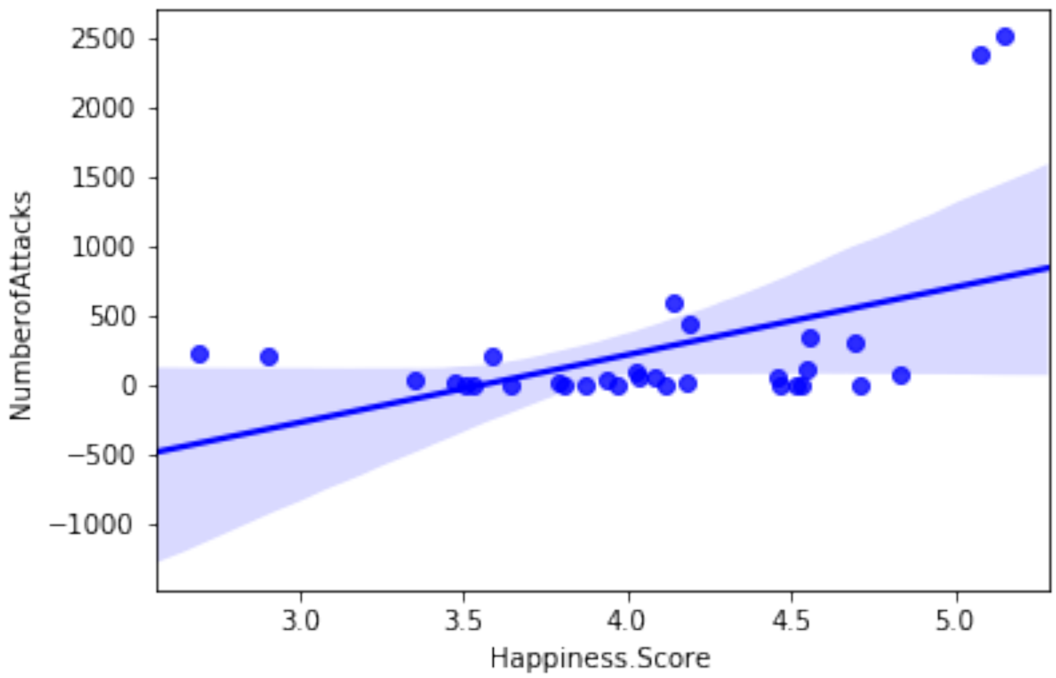


Sub-Saharan Africa Happiness score vs number of attacks

In [13]:

```
SS = df.loc[df['region'] == 11]
SSTerrCnt = SS.country_txt.value_counts()
dfSS = pd.DataFrame(SSTerrCnt)
dfSS = dfSS.rename(columns={'country_txt':'NumberofAttacks'})
dfSS
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfSS],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})


slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is 487
Intercept is -1735
R value is 0.469961
P value is 0.007638
Std Error is 169
```

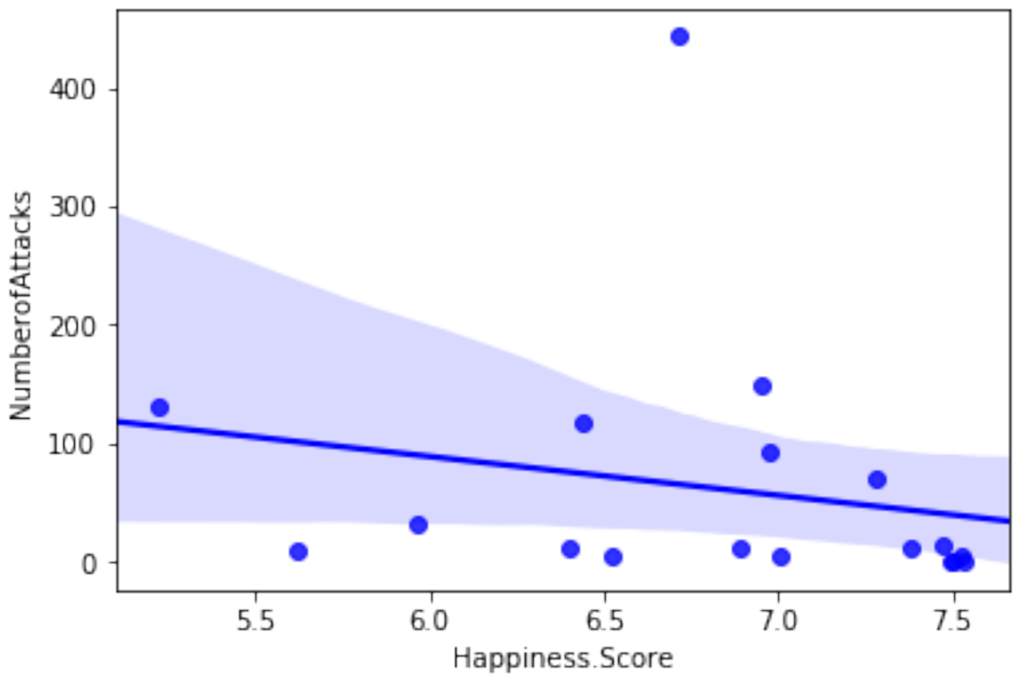


Western Europe Happiness score vs number of attacks

In [14]:

```
WE = df.loc[df['region'] == 8]
WETerrCnt = WE.country_txt.value_counts()
dfWE = pd.DataFrame(WETerrCnt)
dfWE = dfWE.rename(columns={'country_txt':'NumberofAttacks'})
dfWE
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfWE],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})

slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is -32
Intercept is 285
R value is -0.210182
P value is 0.402526
Std Error is 38
```

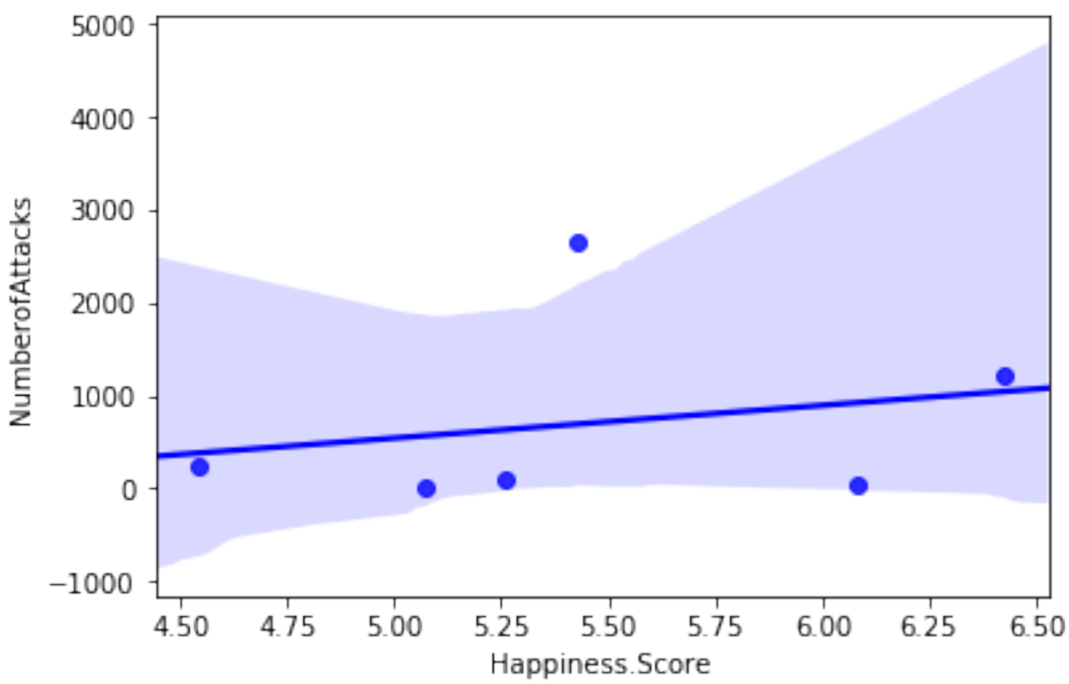


Southeast Asia Happiness score vs number of attacks

In [15]:

```
SEA = df.loc[df['region'] == 5]
SEATerrCnt = SEA.country_txt.value_counts()
dfSEA = pd.DataFrame(SEATerrCnt)
dfSEA = dfSEA.rename(columns={'country_txt':'NumberofAttacks'})
dfSEA
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfSEA],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})


slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is 352
Intercept is -1220
R value is 0.229888
P value is 0.661242
Std Error is 746
```

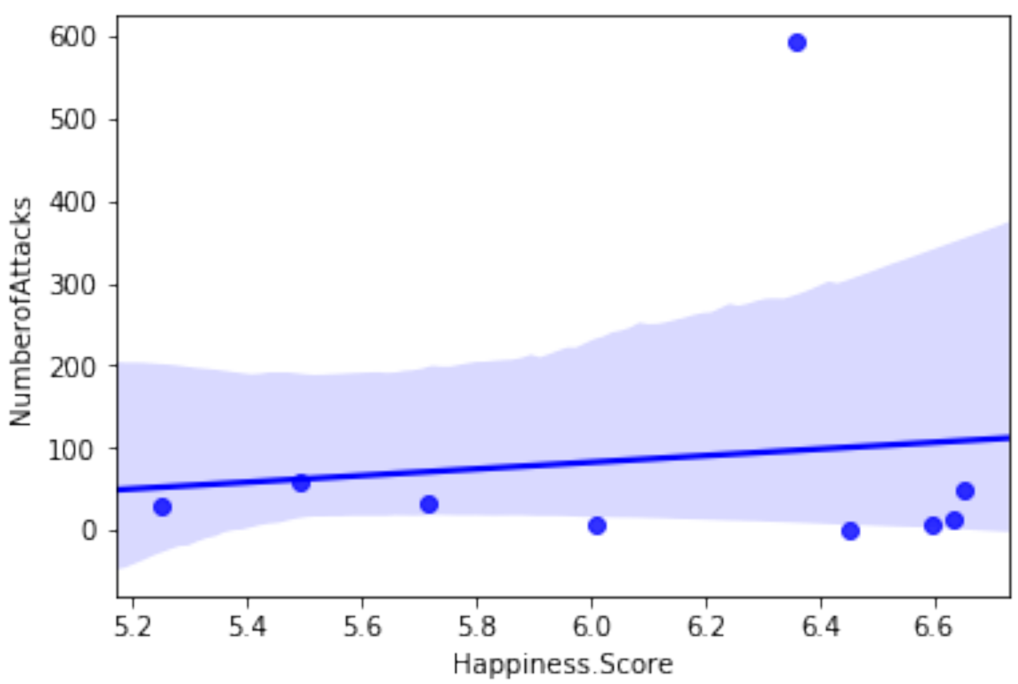


South America Happiness score vs number of attacks

In [16]:

```
SA = df.loc[df['region'] == 3]
SATerrCnt = SA.country_txt.value_counts()
dfSA = pd.DataFrame(SATerrCnt)
dfSA = dfSA.rename(columns={'country_txt':'NumberofAttacks'})
dfSA
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfSA],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})

slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is 40
Intercept is -159
R value is 0.112695
P value is 0.772835
Std Error is 134
```

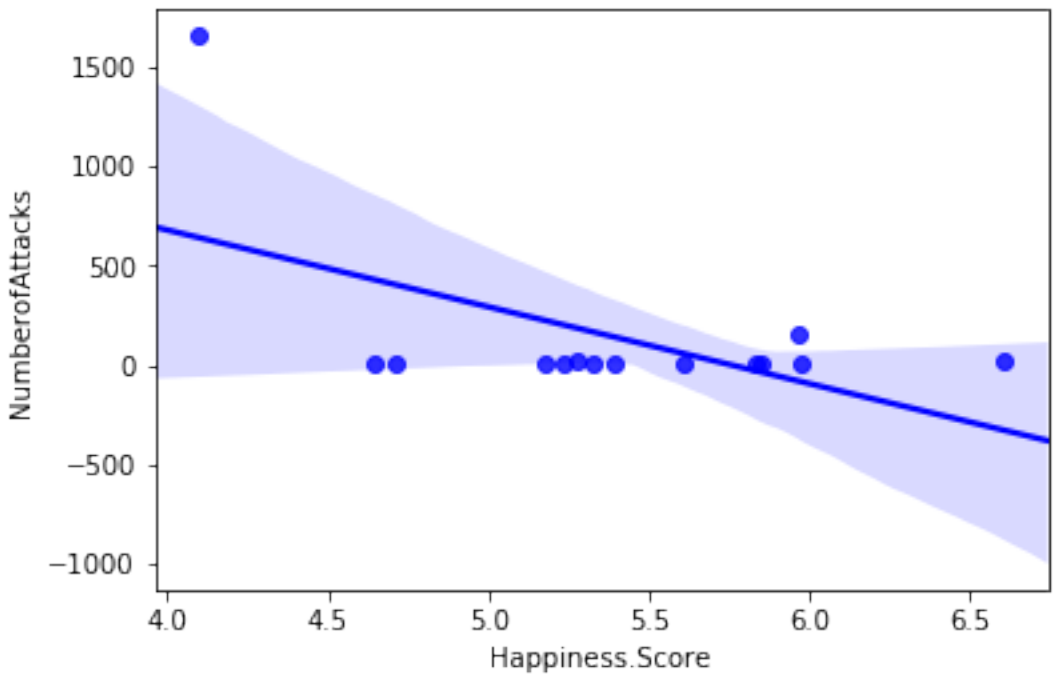


Eastern Europe Happiness score vs number of attacks

In [17]:

```
EE = df.loc[df['region'] == 9]
EETerrCnt = EE.country_txt.value_counts()
dfEE = pd.DataFrame(EETerrCnt)
dfEE = dfEE.rename(columns={'country_txt':'NumberofAttacks'})
dfEE
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')
combined = pd.concat([phappiness,dfEE],axis=1,join='inner')
combined = combined.rename(columns={'country_txt':'NumberofAttacks'})

slope, intercept, r_value, p_value, std_err = stats.linregress(combined['Happine
ss.Score'],combined['NumberofAttacks'])
ax = sns.regplot(x="Happiness.Score", y="NumberofAttacks", data=combined, color=
'b', line_kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
Slope is -385
Intercept is 2220
R value is -0.565177
P value is 0.035195
Std Error is 162
```

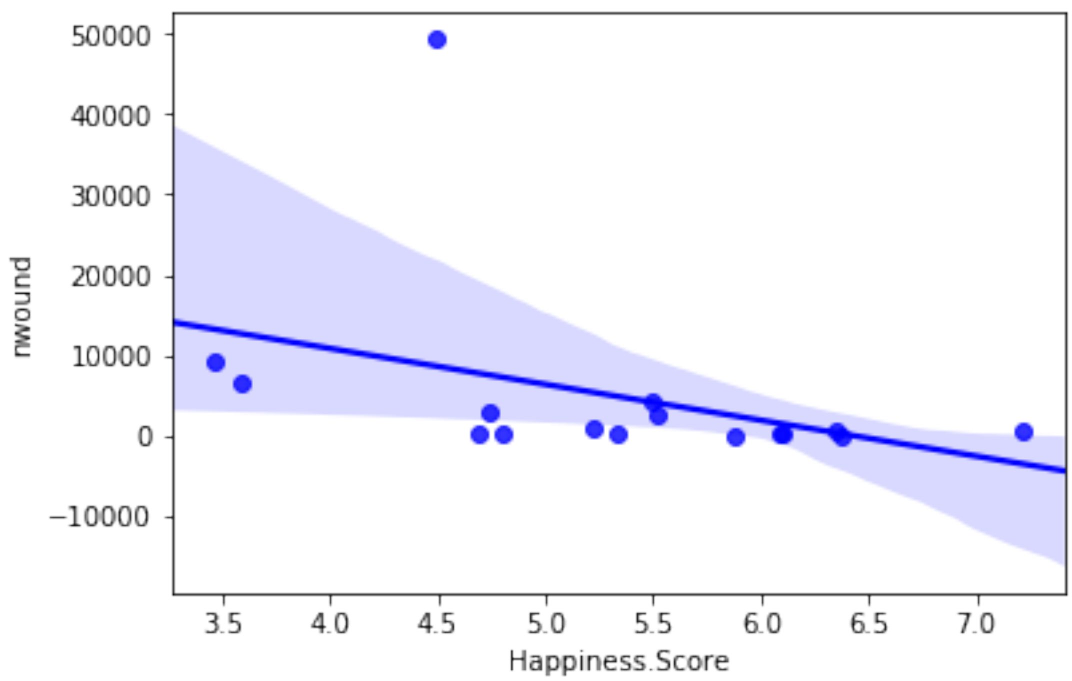


Happiness score vs wounded

In [18]:

```
df = df.loc[df['iyear'] > 2013]
wound = df.loc[df['region'] == 10]
wound = pd.concat([wound['country_txt'],wound['nwound']],axis=1)
filtered = wound.loc[df['nwound'] > 0]
pwound = filtered.set_index('country_txt')
Happiness = pd.concat([happiness['Country'],happiness['Happiness.Score']],axis=1
)
phappiness = Happiness.set_index('Country')

#cant use df because its a different dataset! use Happiness dataset
reducedPwound = pwound.groupby(pwound.index).sum()
wounded = pd.concat([phappiness,reducedPwound],axis=1,join='inner')
print(wounded)
slope, intercept, r_value, p_value, std_err = stats.linregress(wounded['Happines
s.Score'],wounded['nwound'])
ax = sns.regplot(x="Happiness.Score", y="nwound", data=wounded, color='b', line_
kws={'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})

#wounded
print("Slope is %d" % slope)
print("Intercept is %d" % intercept)
print("R value is %f" % r_value)
print("P value is %f" % p_value)
print("Std Error is %d" % std_err)
```

```
              Happiness.Score   nwound
Israel                  7.213    460.0
Qatar                   6.375      1.0
Saudi Arabia            6.344    569.0
Kuwait                  6.105    228.0
Bahrain                 6.087     89.0
Algeria                 5.872     70.0
Libya                   5.525   2651.0
Turkey                  5.500   4292.0
Jordan                  5.336     84.0
Lebanon                 5.225   1002.0
Tunisia                 4.805    216.0
Egypt                   4.735   2854.0
Iran                    4.692     91.0
Iraq                    4.497  49334.0
Yemen                   3.593   6433.0
Syria                   3.462   9277.0
Slope is -4467
Intercept is 28690
R value is -0.372715
P value is 0.155094
Std Error is 2972
```



Happiness Of Nations

In [19]:

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
import matplotlib
```

1. Import Dataset

In [20]:

```
gbterr = df.copy()
#focus on terrorist attacks which happened after 2010 first because that is arou
nd the time the world happiness report gets its data
gbterr2017 = gbterr[gbterr.iyear >2013]
happiness = pd.read_csv('World-Happiness-Report/2017.csv', engine = 'python', in
dex_col=0)
```

2. Visualization

In [21]:

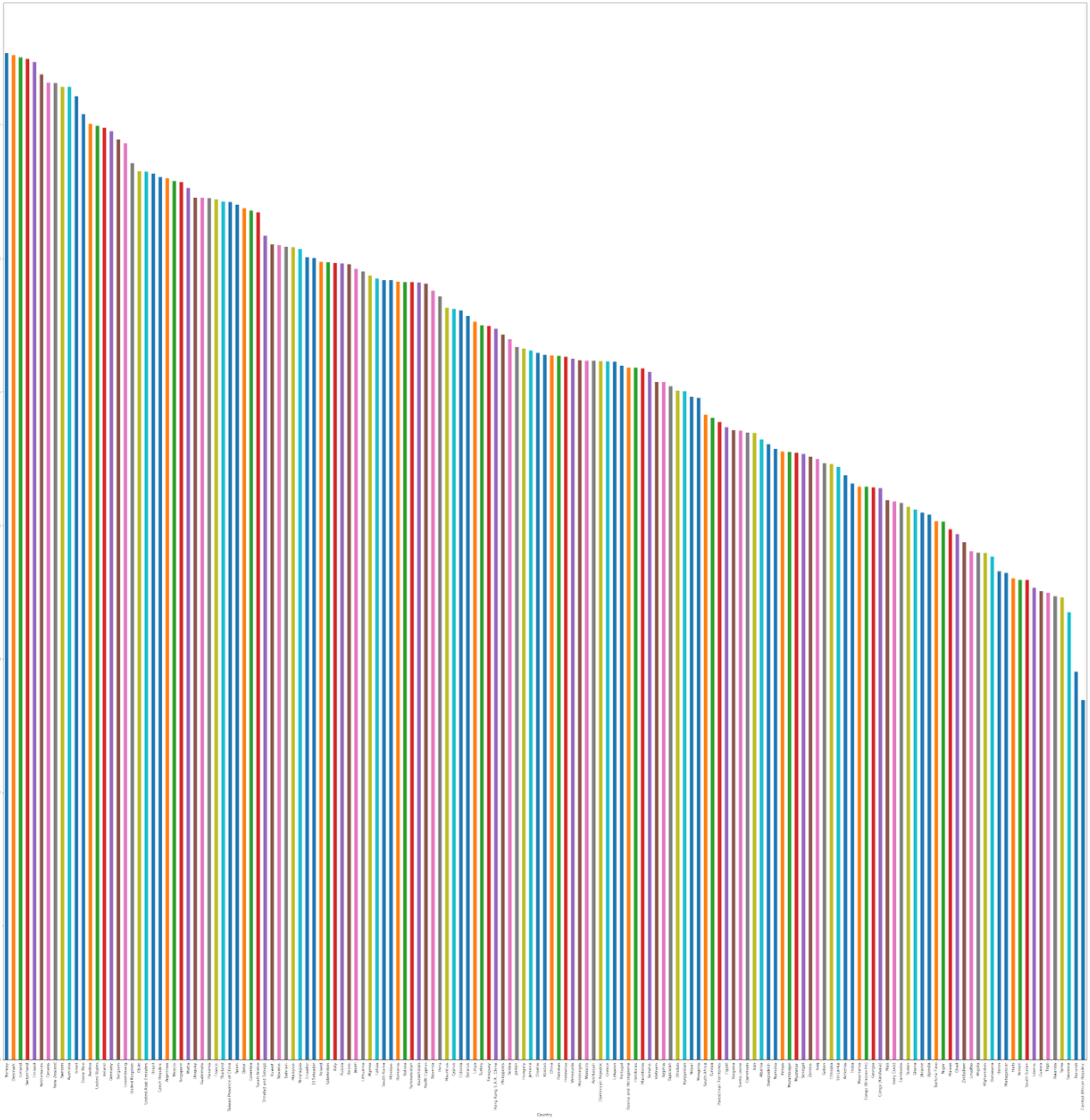
```
#lets visualize happiness scores
mng = plt.get_current_fig_manager()
mng.show_popup("True")
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: Matp
lotlibDeprecationWarning: The show_popup function was deprecated in
version 2.2.
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
<Figure size 432x288 with 0 Axes>
```

In [22]:

```
fig = plt.figure()
fig.set_size_inches(50,50)
happiness['Happiness.Score'].plot('bar')
plt.show()
```

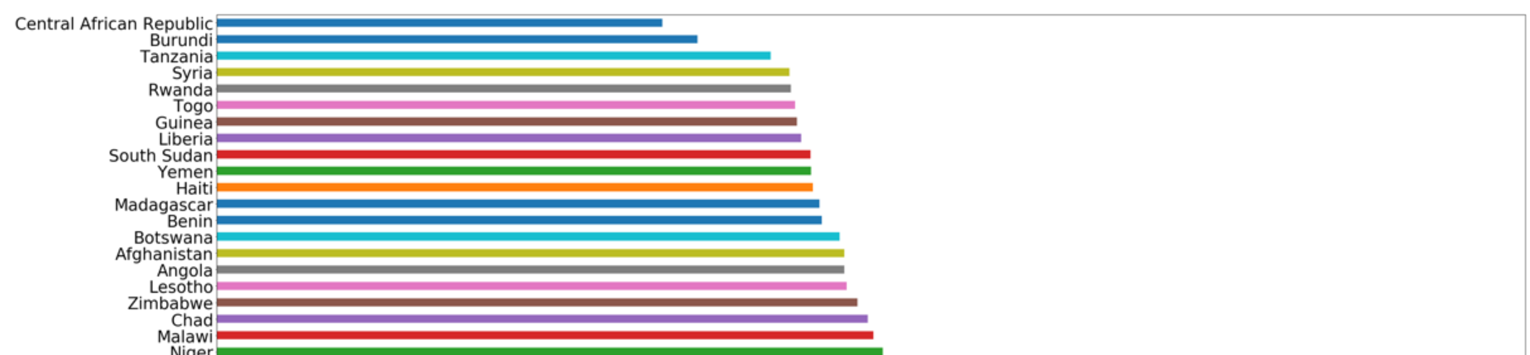


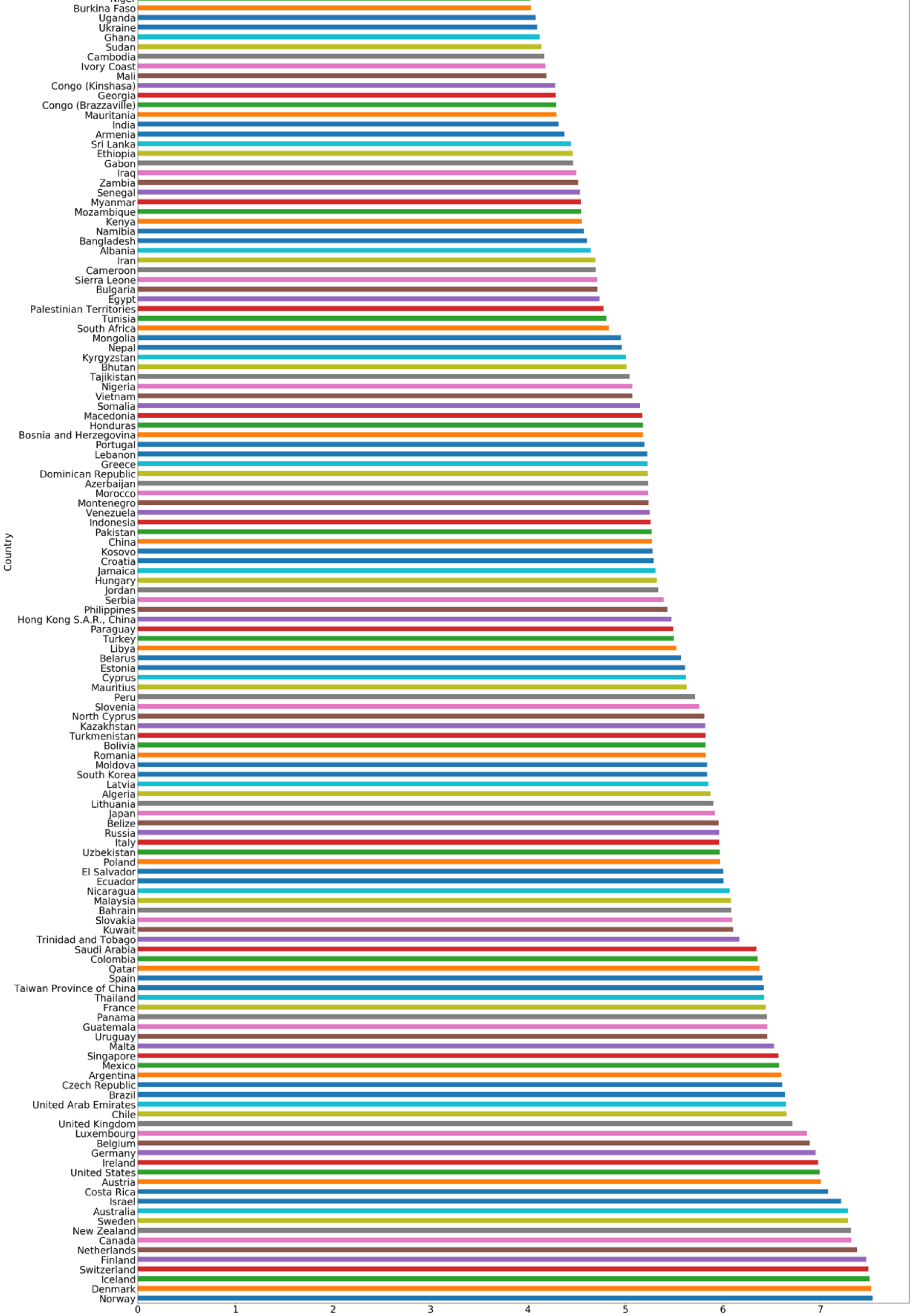
In [23]:

```
matplotlib.rcParams.update({'font.size': 35})
fig = plt.figure()
fig.set_size_inches(50,100)
happiness['Happiness.Score'].plot('barh')
```

Out[23]:

<matplotlib.axes._subplots.AxesSubplot at 0x1c0af55e80>





Niger
Burkina Faso
Uganda
Ukraine
Ghana
Sudan
Cambodia
Ivory Coast
Mali
Congo (Kinshasa)
Georgia
Congo (Brazzaville)
Mauritania
India
Armenia
Sri Lanka
Ethiopia
Gabon
Iraq
Zambia
Senegal
Myanmar
Mozambique
Kenya
Namibia
Bangladesh
Albania
Iran
Cameroon
Sierra Leone
Bulgaria
Egypt
Palestinian Territories
Tunisia
South Africa
Mongolia
Nepal
Kyrgyzstan
Bhutan
Tajikistan
Nigeria
Vietnam
Somalia
Macedonia
Honduras
Bosnia and Herzegovina
Portugal
Lebanon
Greece
Dominican Republic
Azerbaijan
Morocco
Montenegro
Venezuela
Indonesia
Pakistan
China
Kosovo
Croatia
Jamaica
Hungary
Jordan
Serbia
Philippines
Hong Kong S.A.R., China
Paraguay
Turkey
Libya
Belarus
Estonia
Cyprus
Mauritius
Peru
Slovenia
North Cyprus
Kazakhstan
Turkmenistan
Bolivia
Romania
Moldova
South Korea
Latvia
Algeria
Lithuania
Japan
Belize
Russia
Italy
Uzbekistan
Poland
El Salvador
Ecuador
Nicaragua
Malaysia
Bahrain
Slovakia
Kuwait
Trinidad and Tobago
Saudi Arabia
Colombia
Qatar
Spain
Taiwan Province of China
Thailand
France
Panama
Guatemala
Uruguay
Malta
Singapore
Mexico
Argentina
Czech Republic
Brazil
United Arab Emirates
Chile
United Kingdom
Luxembourg
Belgium
Germany
Ireland
United States
Austria
Costa Rica
Israel
Australia
Sweden
New Zealand
Canada
Netherlands
Finland
Switzerland
Iceland
Denmark
Norway
Country
0
1
2
3
4
5
6
7

In [24]:

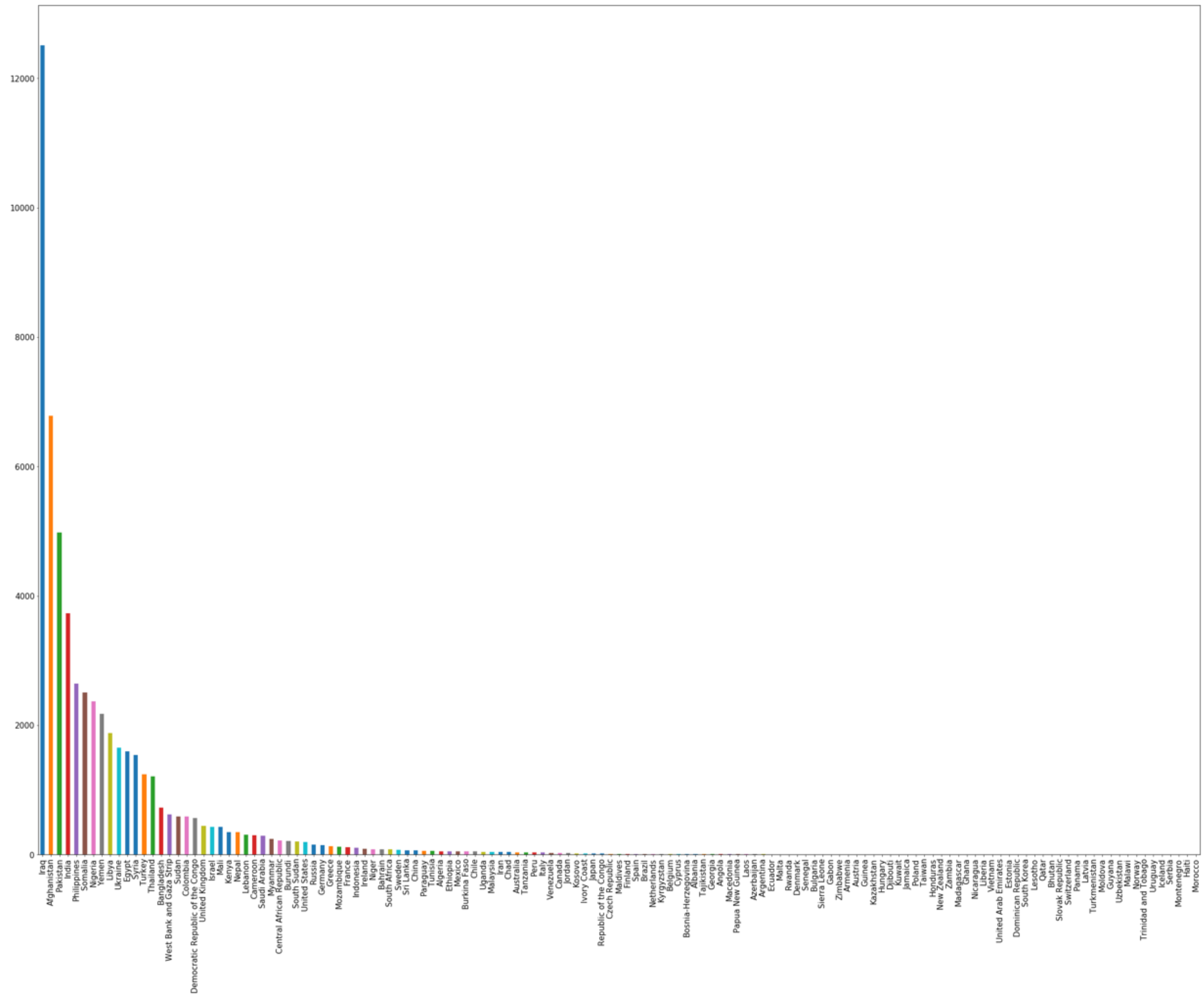
```
#now from the terrorism 2017 dataframe we get the number of attacks per country
tattacks_count = gbterr2017['country_txt'].value_counts()
tattacks_count.head()
```

Out[24]:

```
Iraq            12510
Afghanistan      6783
Pakistan         4977
India            3735
Philippines      2642
Name: country_txt, dtype: int64
```

In [25]:

```
#bar graph of terrorism attacks per country in 2017
fig = plt.figure()
matplotlib.rcParams.update({'font.size': 15})
fig.set_size_inches(40,30)
tattacks_count.plot("bar")
plt.show()
```



3. Correlations

In [26]:

```
#now we look for a correlation between the number terrorist attacks in 2017 in a
country and that country's happiness score as well as other variables in the hap
piness database
#tattacks_count
h_score = pd.Series(happiness['Happiness.Score'])
#h_score.index.name= "Country"
tattacks_count.index.name = "Country"
tattacks_count = pd.DataFrame(tattacks_count)
tattacks_count = tattacks_count.rename(columns={"country_txt":"NumOfAttacks"})
```

In [27]:

```
a_h =  pd.merge(happiness,tattacks_count, on='Country')
dfcorr = a_h[["Happiness.Score", "NumOfAttacks","Economy..GDP.per.Capita.", "Fre
edom","Trust..Government.Corruption.","Dystopia.Residual","Family","Health..Life
.Expectancy." ]]
dfcorr.corr()
```

Out[27]:

	Happiness.Score	NumOfAttacks	Economy..GDP.per.Ca
Happiness.Score	1.000000	-0.169753	0.834426
NumOfAttacks	-0.169753	1.000000	-0.111607
Economy..GDP.per.Capita.	0.834426	-0.111607	1.000000
Freedom	0.619353	-0.169398	0.383448
Trust..Government.Corruption.	0.461512	-0.079760	0.348569
Dystopia.Residual	0.499159	-0.004880	0.109302
Family	0.769330	-0.250197	0.702235
Health..Life.Expectancy.	0.786620	-0.166409	0.834958

In [28]:

```
f,ax = plt.subplots(figsize=(25, 18))
sns.heatmap(dfcorr.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```

Our initial hypothesis was that the number of terrorist attacks a country suffered would have some correlation with that countries hapiness score.

From the correlations shown above, we didn't find substantially large correlations between number of attacks and these variables.

However we did find small negative correlations between the number of attacks and most of the variables that influence happiness in countries.(Except for Dystopia score which had no correlation whatsover). However, the correlations are not as big as we expected in our initial hypothesis.

We decided to look at other data sets in order to find other factors that influence a country's happiness score

UN Country Profiles 2017

This dataset contains key statistical indicators of the countries. It covers 4 major sections

-General Information, Economic Indicators, Social Indicators, Environmental & Infrastructure Indicators

In [29]:

```
un_report =  pd.read_csv('undata-country-profiles/country_profile_variables.csv'
, engine = 'python', index_col= 0)
un_report.index.name = "Country"
un_report.head()
```

Out[29]:

	Region	Surface area (km2)	Population in thousands (2017)	Population density (per km2, 2017)	Sex ratio (m per 100 f, 2017)	GDP: Gross domestic product (million current US$)	GD growt rat (annu: % cons 200 prices
Country							
Afghanistan	SouthernAsia	652864	35530	54.4	106.3	20270	-2.4
Albania	SouthernEurope	28748	2930	106.9	101.9	11541	2.6
Algeria	NorthernAfrica	2381741	41318	17.3	102.0	164779	3.8
American Samoa	Polynesia	199	56	278.2	103.6	-99	-99
Andorra	SouthernEurope	468	77	163.8	102.3	2812	0.8

5 rows × 49 columns

In order to better visualize our data we decided to group countries into groups according to their GDP

We will be using the World Bank's classfication of low income, lower middle income, upper middle income, and high-income.

#Countries with less than $1,035 GNI per capita are classified as low-income countries, those with between$1,036 and $4,085 as lower middle income countries, those with between$4,086 and $12,615 as upper middle income countries, and those with incomes of more than$12,615 as high-income countries.

In [30]:

```
un_report["GDP per capita (current US$)"].head()
```

Out[30]:

```
Country
Afghanistan          623.2
Albania             3984.2
Algeria             4154.1
American Samoa       -99.0
Andorra            39896.4
Name: GDP per capita (current US$), dtype: float64
```

Low-Income Countries (GDP per capita < 1035)

In [31]:

```
low_income = un_report[un_report["GDP per capita (current US$)"] < 1035]
li_gdpc = low_income[low_income["GDP per capita (current US$)"] >0]
li_gdpc  = li_gdpc["GDP per capita (current US$)"]
li_gdpc = li_gdpc.sort_values()
```

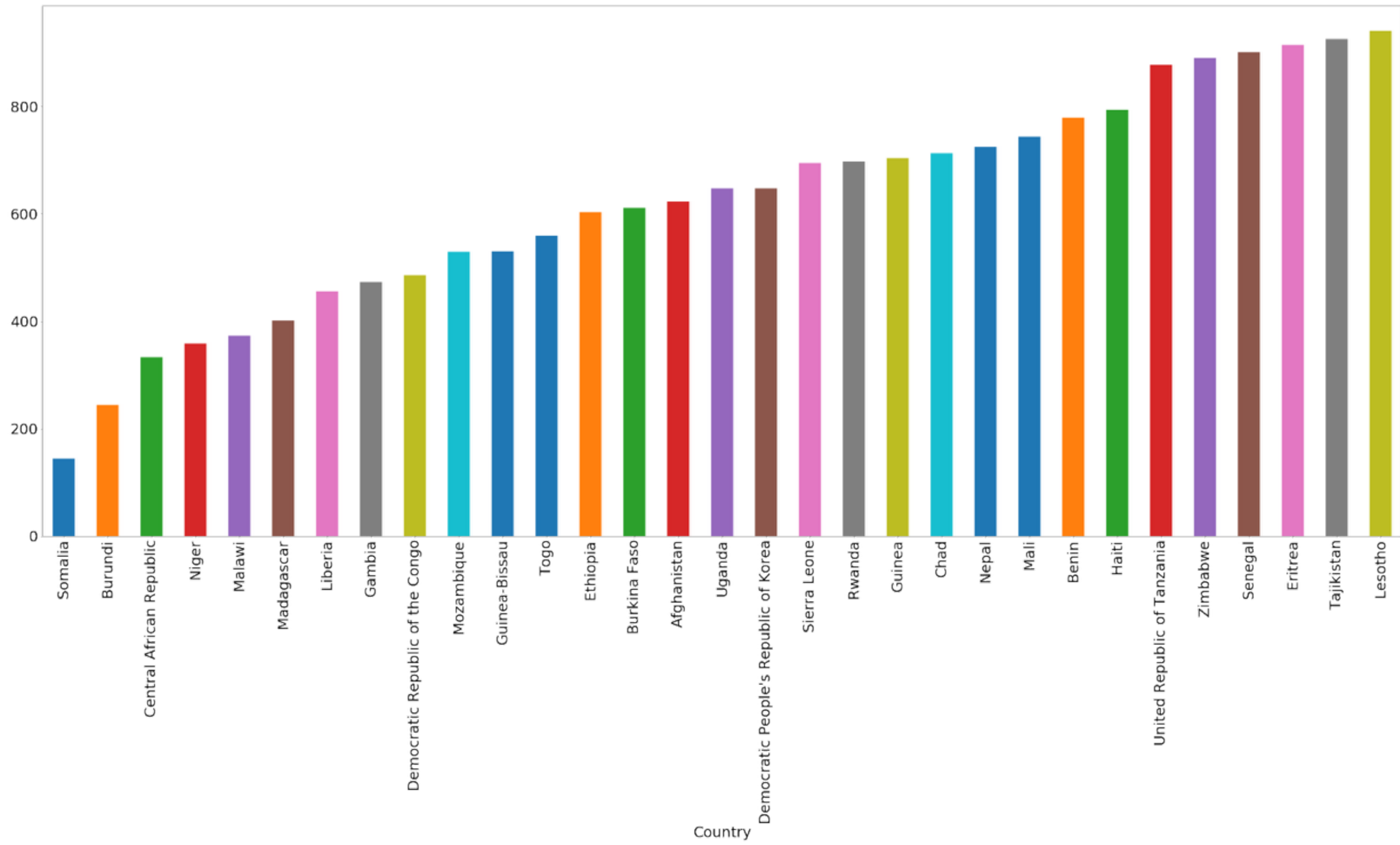
Bar Graph: Low Income countries and their GDP

In [32]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,20)
li_gdpc.plot("bar")
```

Out[32]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x1c0af61f60&gt;
```



Happiness Scores of Low Income Countries

In [33]:

```
li_gdpc.index
h_score.index
h_score_lowincome = h_score[li_gdpc.index]
h_score_lowincome = h_score_lowincome.dropna()
h_score_lowincome = h_score_lowincome.sort_values()
h_score_lowincome.head()
```

Out[33]:

```
Country
Central African Republic    2.693
Burundi                     2.905
Rwanda                      3.471
Togo                        3.495
Guinea                      3.507
Name: Happiness.Score, dtype: float64
```

In [34]:

```
h_score_lowincome
```

Out[34]:

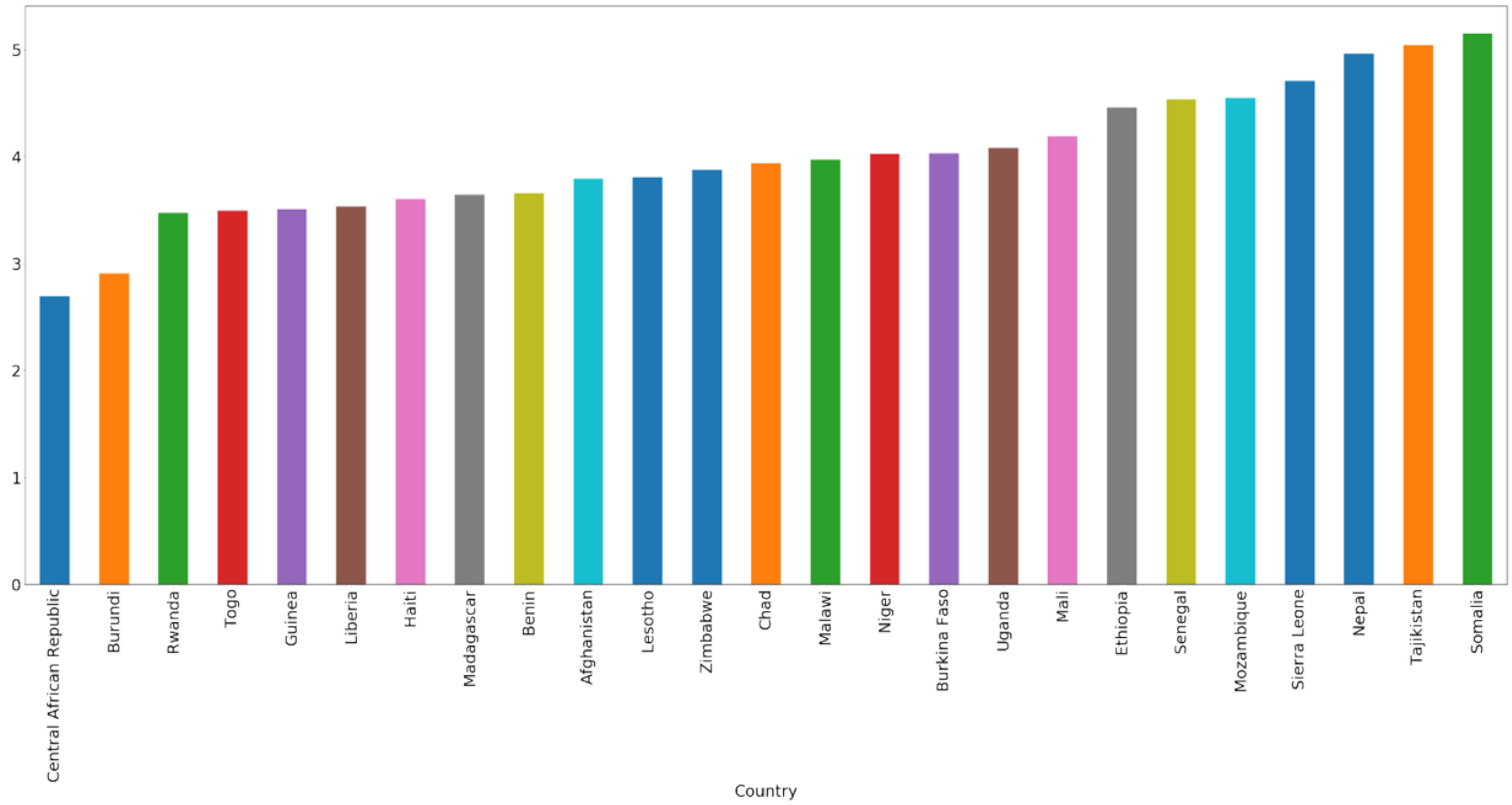
```
Country
Central African Republic    2.693
Burundi                     2.905
Rwanda                      3.471
Togo                        3.495
Guinea                      3.507
Liberia                     3.533
Haiti                       3.603
Madagascar                  3.644
Benin                       3.657
Afghanistan                 3.794
Lesotho                     3.808
Zimbabwe                    3.875
Chad                        3.936
Malawi                      3.970
Niger                       4.028
Burkina Faso                4.032
Uganda                      4.081
Mali                        4.190
Ethiopia                    4.460
Senegal                     4.535
Mozambique                  4.550
Sierra Leone                4.709
Nepal                       4.962
Tajikistan                  5.041
Somalia                     5.151
Name: Happiness.Score, dtype: float64
```

In [35]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,20)
h_score_lowincome.plot("bar")
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0b09b3c8>
```



In [36]:

```
h_score_lowincome.describe()
```

Out[36]:

```
count    25.000000
mean      3.985200
std       0.615126
min       2.693000
25%       3.603000
50%       3.936000
75%       4.460000
max       5.151000
Name: Happiness.Score, dtype: float64
```

Terrorist Attacks in Low Income Countries

In [37]:

```
tattacks_count = gbterr2017['country_txt'].value_counts()
tattacks_count.index.name = "Country"
tattacks_count.name = "Terrorist Attacks Count"
```

In [38]:

```
tattacks_lowincome = tattacks_count[h_score_lowincome.index]
tattacks_lowincome = tattacks_lowincome.dropna()
tattacks_lowincome = tattacks_lowincome.sort_values()
tattacks_lowincome.head()
```

Out[38]:

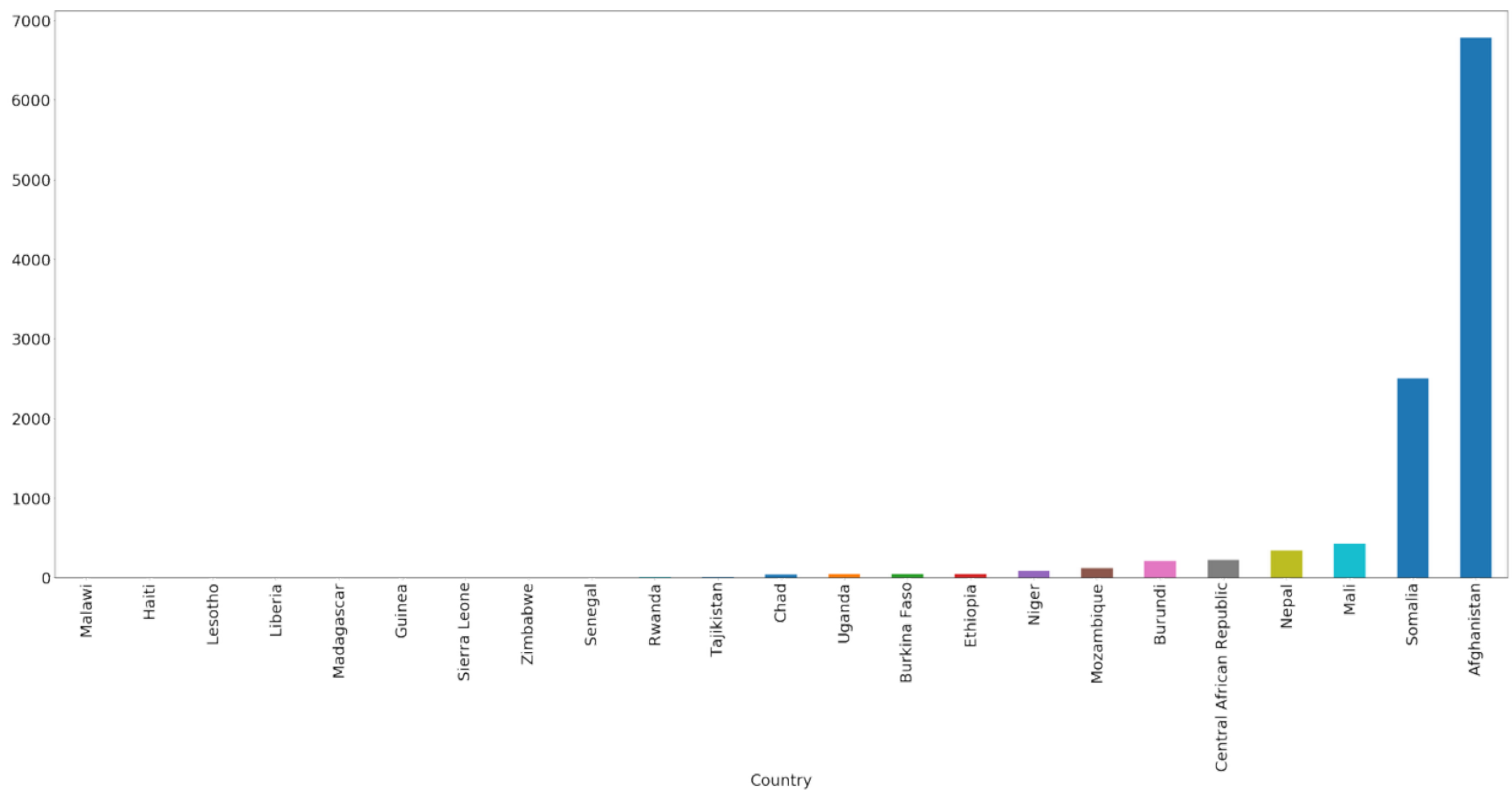
```
Country
Malawi        1.0
Haiti         1.0
Lesotho       1.0
Liberia       2.0
Madagascar    2.0
Name: Terrorist Attacks Count, dtype: float64
```

In [39]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,20)
tattacks_lowincome.plot("bar")
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0b5e6cf8>
```



In [40]:

```
tattacks_lowincome.describe()
```

Out[40]:

```
count      23.000000
mean      474.652174
std      1469.930382
min         1.000000
25%         3.500000
50%        39.000000
75%       165.500000
max      6783.000000
Name: Terrorist Attacks Count, dtype: float64
```

Relationship between the number Terrorism Attacks and Happiness Score of Low Income Countries

In [41]:

```
tattacks_lowincome.corr(h_score_lowincome)
```

Out[41]:

```
0.06151287918229158
```

Relationship between the number Terrorism Attacks and Happiness Score other hapiness variables of Low Income Countries

In [42]:

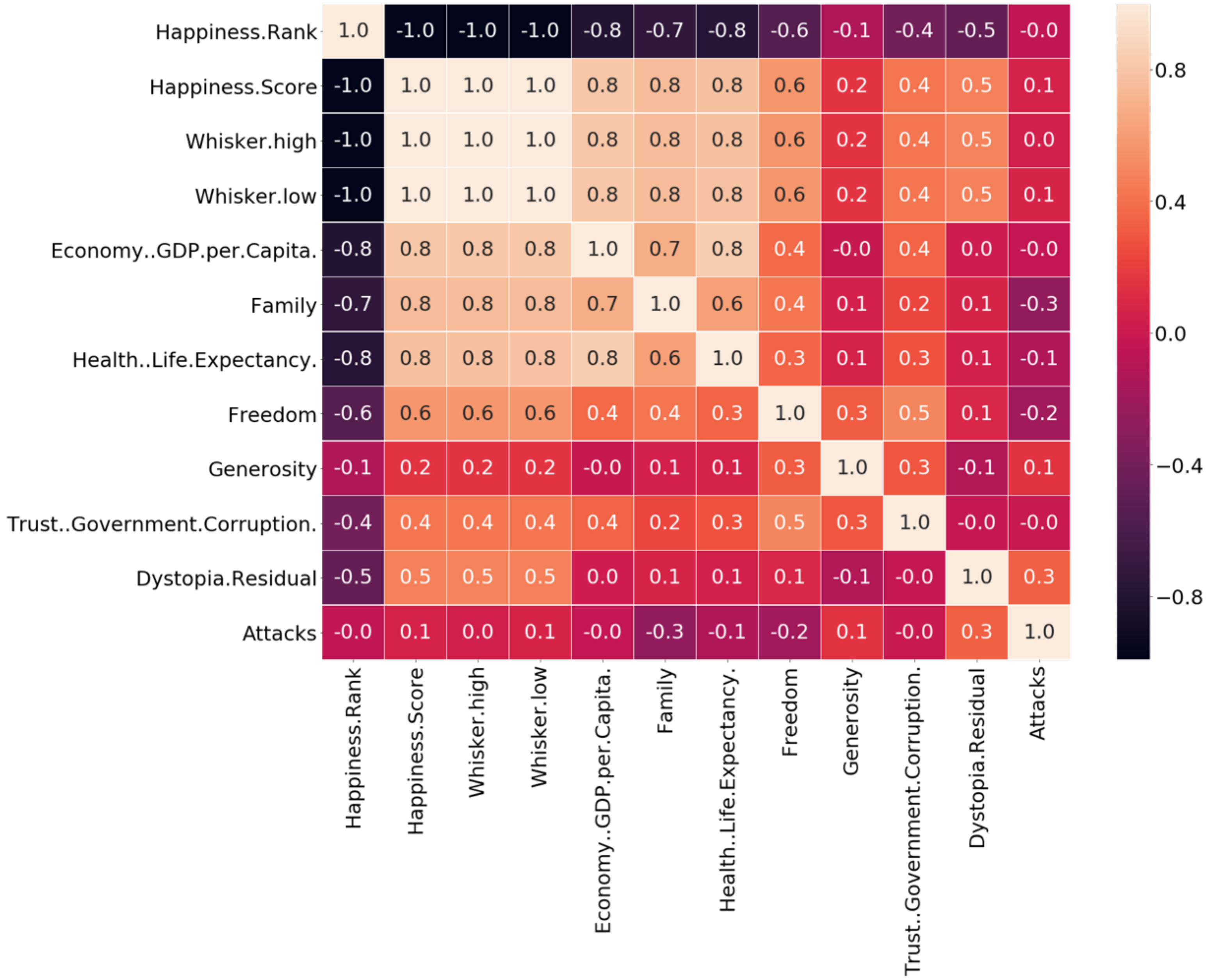
```
lowincomea_h = happiness.copy()
lowincomea_h["Attacks"] = tattacks_lowincome
lowincomea_h.corr()
```

Out[42]:

	Happiness.Rank	Happiness.Score	Whisker.high	Whis
Happiness.Rank	1.000000	-0.992774	-0.993058	-0.99
Happiness.Score	-0.992774	1.000000	0.999497	0.999
Whisker.high	-0.993058	0.999497	1.000000	0.998
Whisker.low	-0.991533	0.999520	0.998036	1.000
Economy..GDP.per.Capita.	-0.813244	0.812469	0.811868	0.812
Family	-0.736753	0.752737	0.750934	0.753
Health..Life.Expectancy.	-0.780716	0.781951	0.776634	0.786
Freedom	-0.551608	0.570137	0.569907	0.569
Generosity	-0.132620	0.155256	0.155462	0.154
Trust..Government.Corruption.	-0.405842	0.429080	0.426459	0.431
Dystopia.Residual	-0.484506	0.475355	0.478824	0.471
Attacks	-0.047867	0.061513	0.049375	0.072

In [43]:

```
f,ax = plt.subplots(figsize=(25, 18))
sns.heatmap(lowincomea_h.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



	Happiness.Rank	Happiness.Score	Whisker.high	Whisker.low	Economy..GDP.per.Capita.	Family	Health..Life.Expectancy.	Freedom	Generosity	Trust..Government.Corruption.	Dystopia.Residual	Attacks
Happiness.Rank	1.0	-1.0	-1.0	-1.0	-0.8	-0.7	-0.8	-0.6	-0.1	-0.4	-0.5	-0.0
Happiness.Score	-1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.6	0.2	0.4	0.5	0.1
Whisker.high	-1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.6	0.2	0.4	0.5	0.0
Whisker.low	-1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.6	0.2	0.4	0.5	0.1
Economy..GDP.per.Capita.	-0.8	0.8	0.8	0.8	1.0	0.7	0.8	0.4	-0.0	0.4	0.0	-0.0
Family	-0.7	0.8	0.8	0.8	0.7	1.0	0.6	0.4	0.1	0.2	0.1	-0.3
Health..Life.Expectancy.	-0.8	0.8	0.8	0.8	0.8	0.6	1.0	0.3	0.1	0.3	0.1	-0.1
Freedom	-0.6	0.6	0.6	0.6	0.4	0.4	0.3	1.0	0.3	0.5	0.1	-0.2
Generosity	-0.1	0.2	0.2	0.2	-0.0	0.1	0.1	0.3	1.0	0.3	-0.1	0.1
Trust..Government.Corruption.	-0.4	0.4	0.4	0.4	0.4	0.2	0.3	0.5	0.3	1.0	-0.0	-0.0
Dystopia.Residual	-0.5	0.5	0.5	0.5	0.0	0.1	0.1	0.1	-0.1	-0.0	1.0	0.3
Attacks	-0.0	0.1	0.0	0.1	-0.0	-0.3	-0.1	-0.2	0.1	-0.0	0.3	1.0

Lower Middle Income Countries (GDP per Capita between 1,036 and 4,085)

In [44]:

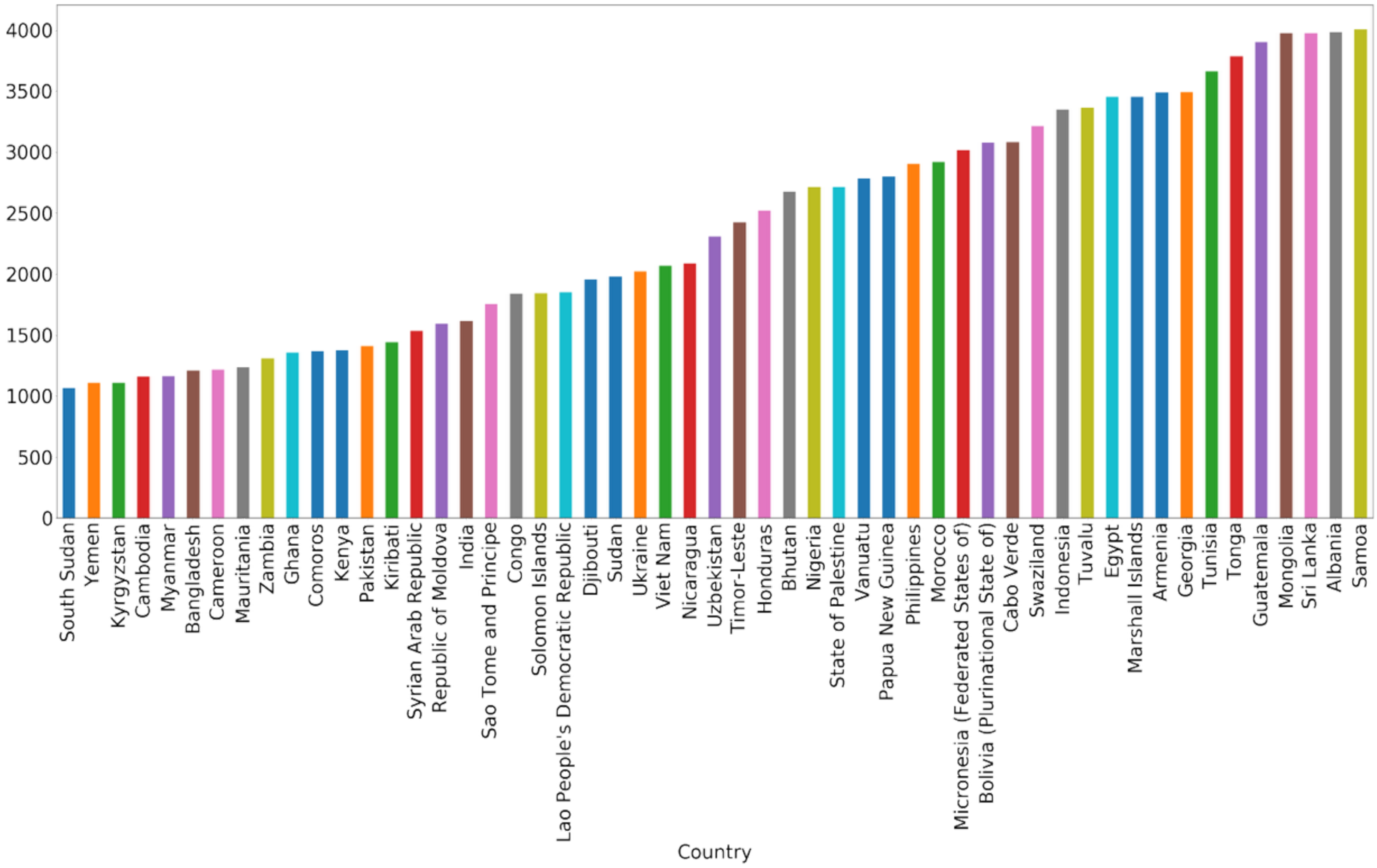
```
low_middle_income = un_report[un_report["GDP per capita (current US$)"] > 1035
]
low_middle_income = low_middle_income[low_middle_income["GDP per capita (current
US$)"] <4085]
lmi_gdpc = low_middle_income["GDP per capita (current US$)"]
lmi_gdpc = lmi_gdpc.sort_values()
matplotlib.rcParams.update({'font.size': 40})
```

In [45]:

```
fig = plt.figure()
fig.set_size_inches(50,20)
lmi_gdpc.plot("bar")
```

Out[45]:

<matplotlib.axes._subplots.AxesSubplot at 0x1c0b6619b0>



In [46]:

```
lmi_gdpc.describe()
```

Out[46]:

```
count      53.000000
mean     2390.367925
std       961.348547
min      1067.000000
25%      1442.900000
50%      2308.300000
75%      3211.700000
max      4006.000000
Name: GDP per capita (current US$), dtype: float64
```

Happiness Scores of Lower Middle Income Countries

In [47]:

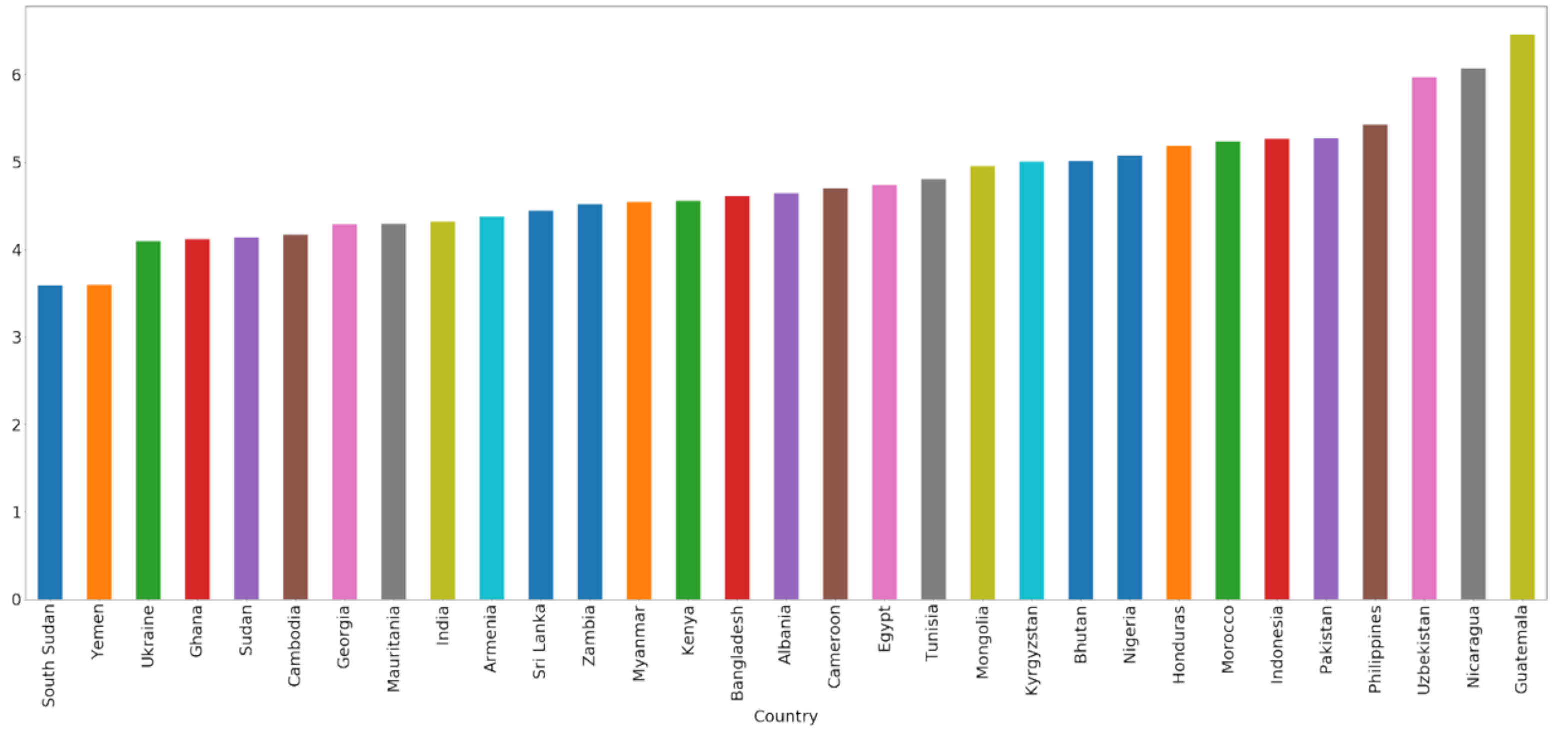
```
matplotlib.rcParams.update({'font.size': 10})
lmi_hscore = h_score[lmi_gdpc.index]
lmi_hscore = lmi_hscore.sort_values()
lmi_hscore =lmi_hscore.dropna()
lmi_hscore.head()
lmi_hscore.name= "Happiness Score"
```

In [48]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,20)
lmi_hscore.plot("bar")
```

Out[48]:

<matplotlib.axes._subplots.AxesSubplot at 0x1c0b7e3048>



In [49]:

```
lmi_hscore.describe()
```

Out[49]:

```
count    31.000000
mean      4.755871
std       0.660202
min       3.591000
25%       4.303500
50%       4.644000
75%       5.127500
max       6.454000
Name: Happiness Score, dtype: float64
```

Terrorist Attacks in Lower Middle Income Countries

In [50]:

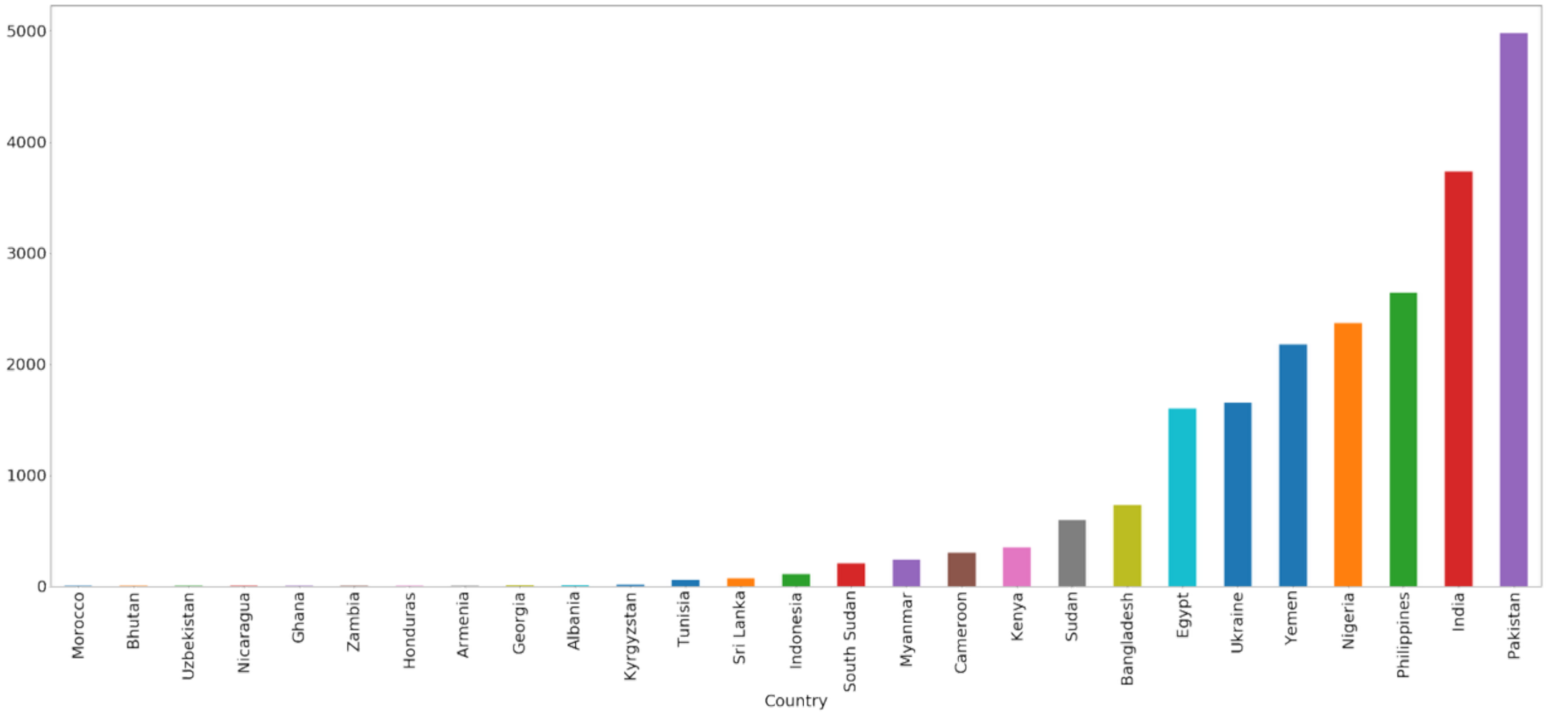
```
tattacks_lmi = tattacks_count[lmi_hscore.index]
tattacks_lmi = tattacks_lmi.dropna()
tattacks_lmi =tattacks_lmi.sort_values()
```

In [51]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,20)
tattacks_lmi.plot("bar")
```

Out[51]:

<matplotlib.axes._subplots.AxesSubplot at 0x1c0af9c4a8>



In [52]:

```
tattacks_lmi.describe()
```

Out[52]:

```
count       27.000000
mean       809.407407
std       1313.870508
min          1.000000
25%          3.500000
50%        110.000000
75%       1164.500000
max       4977.000000
Name: Terrorist Attacks Count, dtype: float64
```

Relationship between the number Terrorism Attacks and Happiness Score of Lower Middle Income Countries

In [53]:

```
tattacks_lmi.head()
```

Out[53]:

```
Country
Morocco       1.0
Bhutan        1.0
Uzbekistan    1.0
Nicaragua     2.0
Ghana         2.0
Name: Terrorist Attacks Count, dtype: float64
```

In [54]:

```
lmi_hscore.head()
```

Out[54]:

```
Country
South Sudan    3.591
Yemen          3.593
Ukraine        4.096
Ghana          4.120
Sudan          4.139
Name: Happiness Score, dtype: float64
```

In [55]:

```
lmi_hscore.corr(tattacks_lmi)
```

Out[55]:

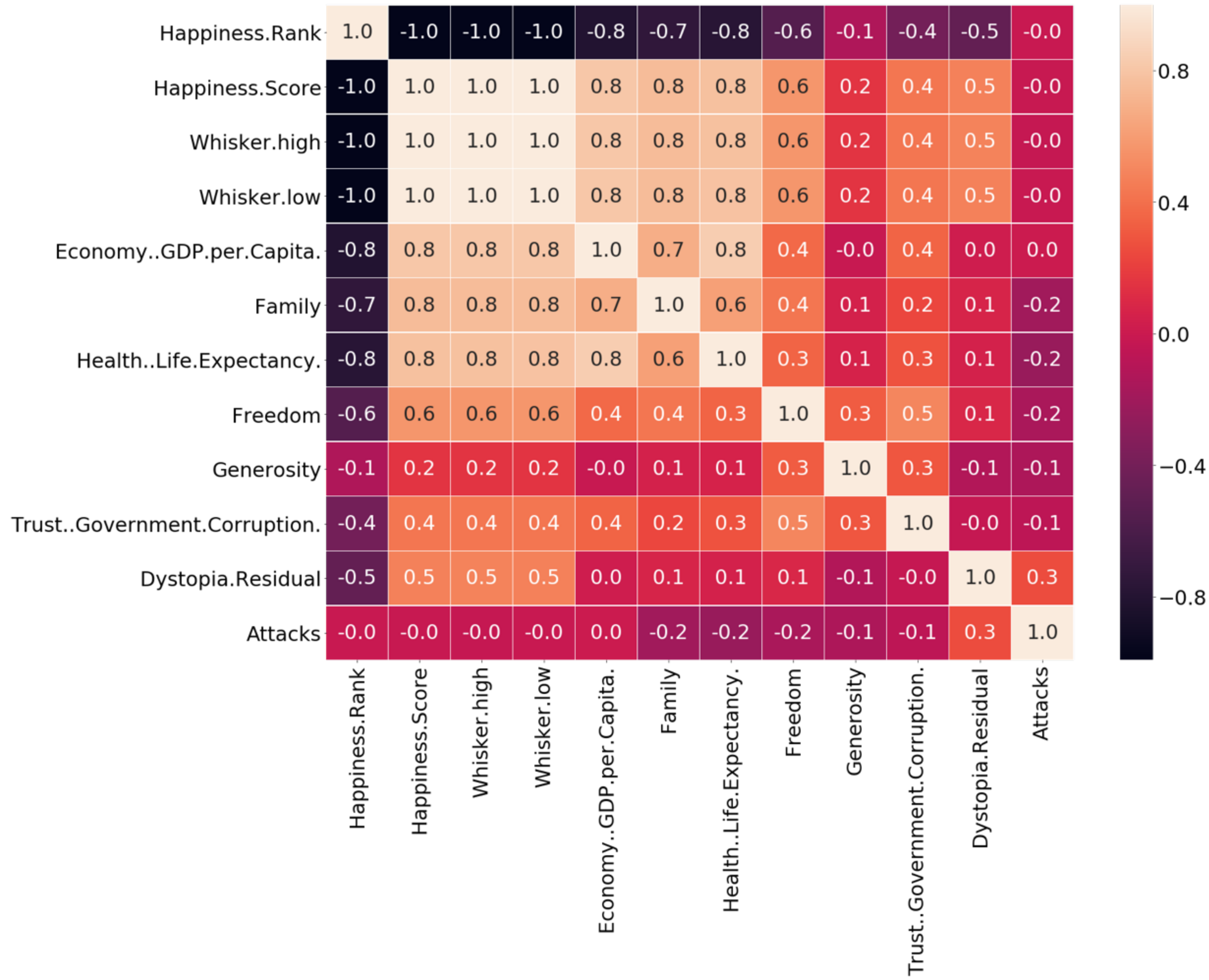
```
-0.016847750282220065
```

In [56]:

```
lowerMiddleincomea_h = happiness.copy()
lowerMiddleincomea_h["Attacks"] = tattacks_lmi
```

In [57]:

```
f,ax = plt.subplots(figsize=(25, 18))
sns.heatmap(lowerMiddleincomea_h.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



Upper Middle Income Countries (GDP per capita 4,086 and 12,615)

In [58]:

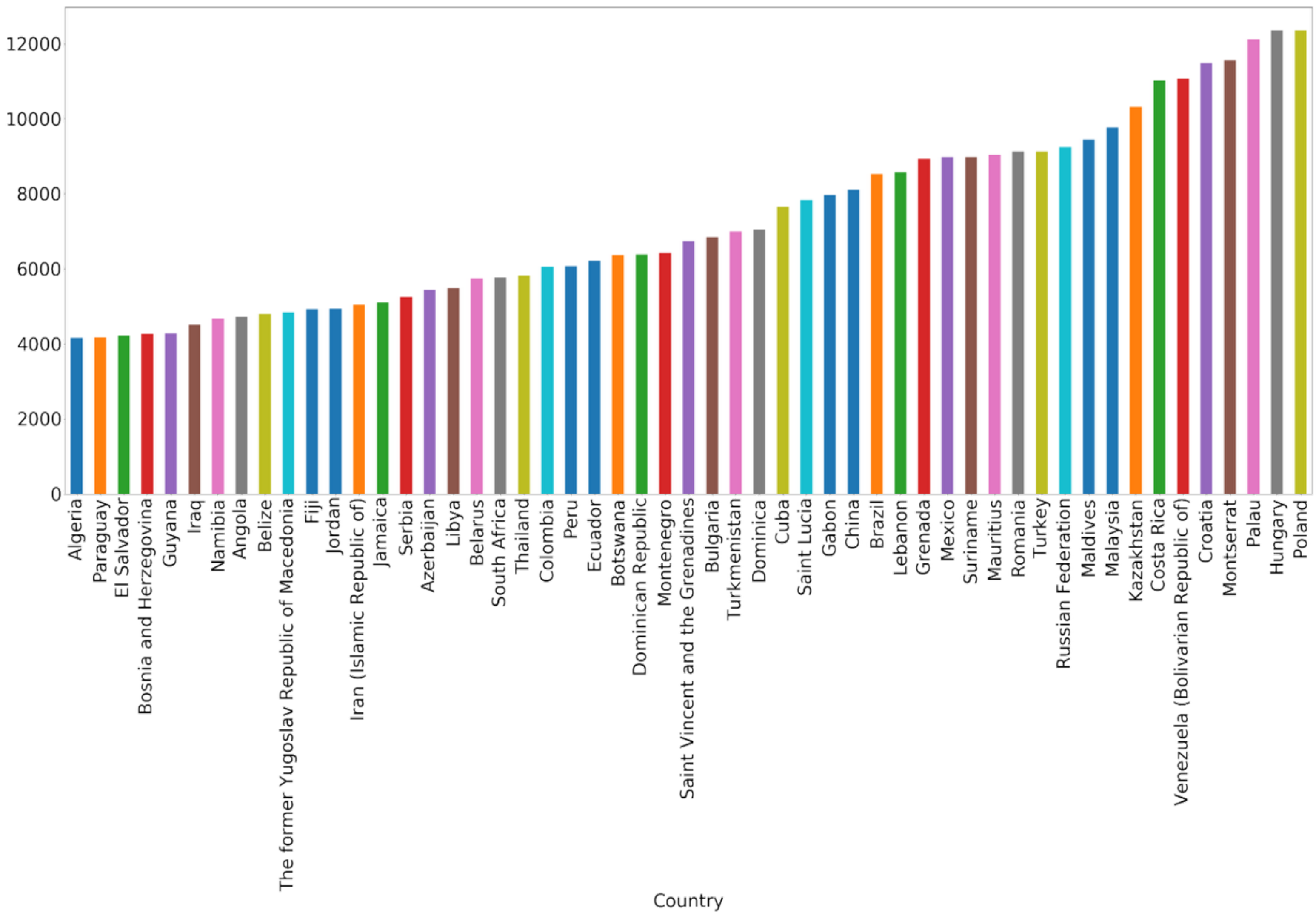
```
upper_middle_income = un_report[un_report["GDP per capita (current US$)"] > 4086]
upper_middle_income = upper_middle_income[upper_middle_income["GDP per capita (current US$)"] <12615]
umi_gdpc = upper_middle_income["GDP per capita (current US$)"]
umi_gdpc = umi_gdpc.sort_values()
matplotlib.rcParams.update({'font.size': 40})
```

In [59]:

```
fig = plt.figure()
fig.set_size_inches(50,20)
umi_gdpc.plot("bar")
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0aebdbe0>
```



In [60]:

```
umi_gdpc.describe()
```

Out[60]:

```
count       53.000000
mean      7298.392453
std       2448.946058
min       4154.100000
25%       5105.800000
50%       6739.200000
75%       9040.900000
max      12355.500000
Name: GDP per capita (current US$), dtype: float64
```

In [61]:

```
matplotlib.rcParams.update({'font.size': 20})
```

Happiness Scores of Upper Middle Income Countries

In [62]:

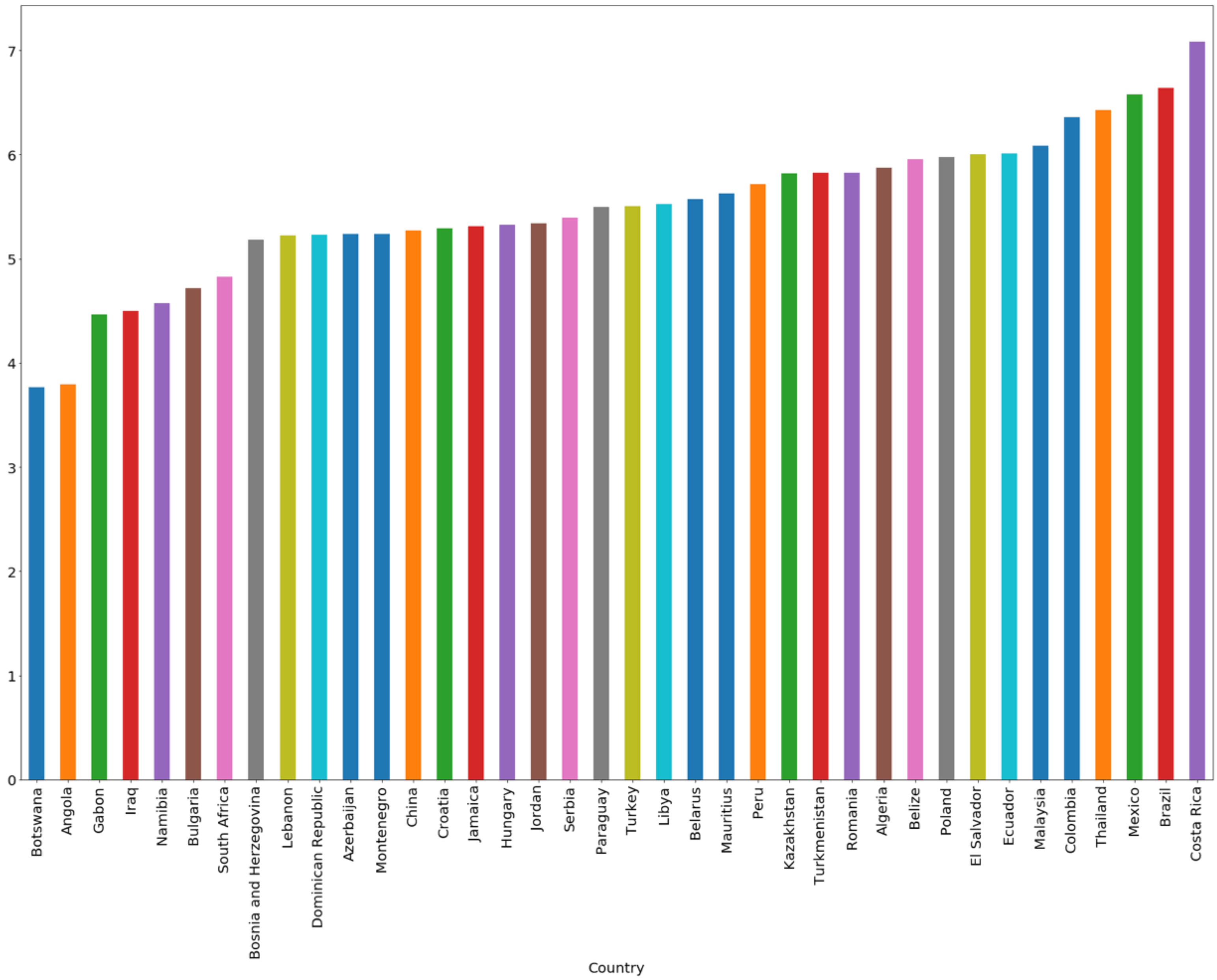
```
umi_hscore = h_score[umi_gdpc.index]
umi_hscore = umi_hscore.dropna()
umi_hscore = umi_hscore.sort_values()
```

In [63]:

```
fig = plt.figure()
fig.set_size_inches(30,20)
umi_hscore.plot("bar")
```

Out[63]:

<matplotlib.axes._subplots.AxesSubplot at 0x1c0adc9908>



In [64]:

```
umi_hscore.describe()
```

Out[64]:

```
count    38.000000
mean      5.488053
std       0.713442
min       3.766000
25%       5.231000
50%       5.496500
75%       5.935000
max       7.079000
Name: Happiness.Score, dtype: float64
```

Terrorist Attacks in Lower Middle Income Countries

In [65]:

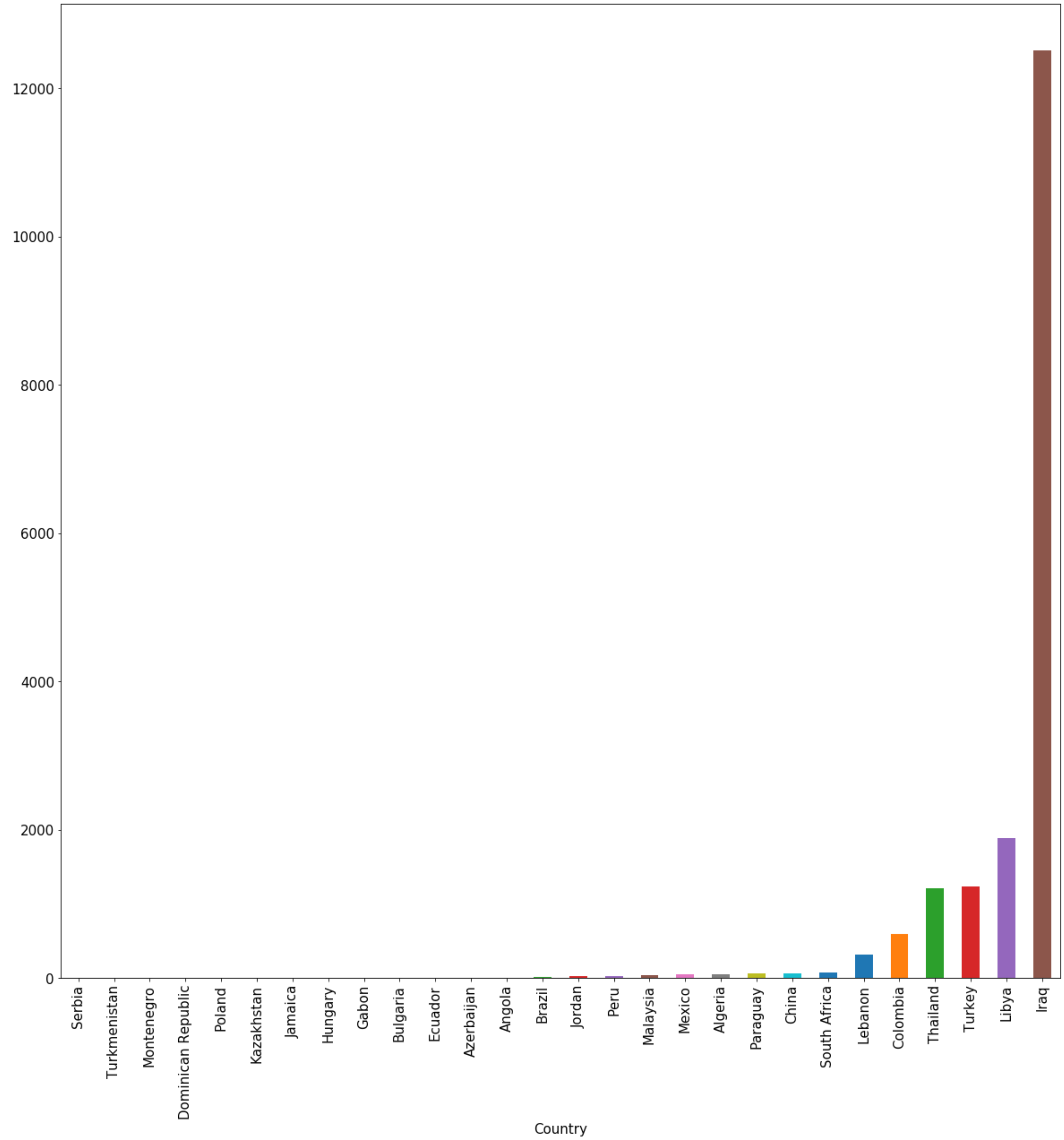
```
tattacks_umi = tattacks_count[umi_hscore.index]
tattacks_umi = tattacks_umi.dropna()
tattacks_umi = tattacks_umi.sort_values()
```

In [66]:

```
matplotlib.rcParams.update({'font.size': 15})
fig = plt.figure()
fig.set_size_inches(20,20)
tattacks_umi.plot("bar")
```

Out[66]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0b8b3550>
```



In [67]:

```
tattacks_umi.describe()
```

Out[67]:

```
count       28.000000
mean       650.250000
std       2370.149919
min          1.000000
25%          3.000000
50%         18.000000
75%         69.000000
max      12510.000000
Name: Terrorist Attacks Count, dtype: float64
```

Relationship between the number Terrorism Attacks and Happiness Score of Upper Middle Income Countries

In [68]:

```
tattacks_umi.head()
```

Out[68]:

```
Country
Serbia                1.0
Turkmenistan          1.0
Montenegro            1.0
Dominican Republic    2.0
Poland                3.0
Name: Terrorist Attacks Count, dtype: float64
```

In [69]:

```
umi_hscore.head()
```

Out[69]:

```
Country
Botswana    3.766
Angola      3.795
Gabon       4.465
Iraq        4.497
Namibia     4.574
Name: Happiness.Score, dtype: float64
```

In [70]:

```
tattacks_umi.corr(umi_hscore)
```

Out[70]:

```
-0.25404708132522136
```

In [71]:

```
upperMiddleincomea_h = happiness.copy()
upperMiddleincomea_h["Attacks"] = tattacks_umi
upperMiddleincomea_h = upperMiddleincomea_h.dropna()
upperMiddleincomea_h
```

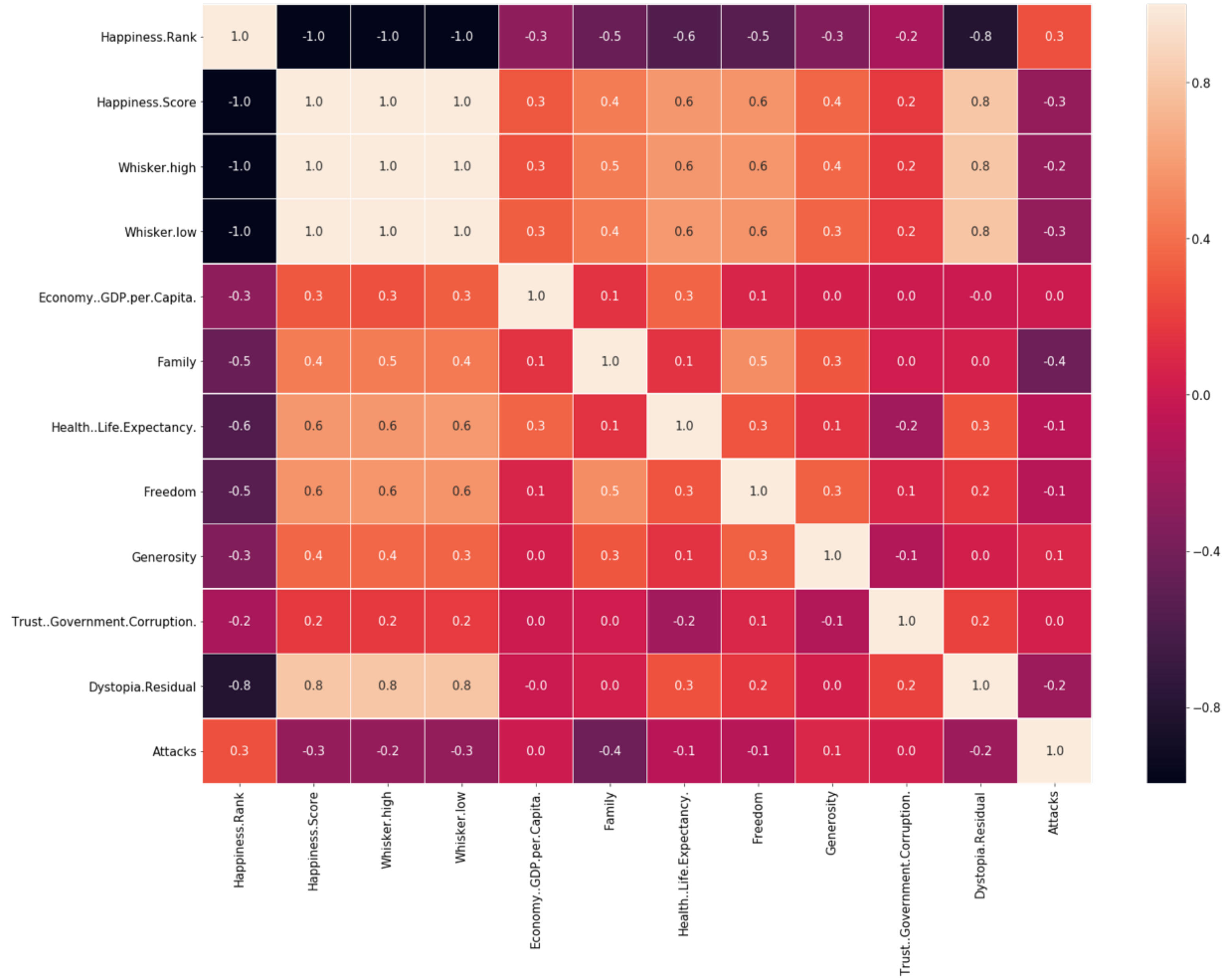
Out[71]:

	Happiness.Rank	Happiness.Score	Whisker.high	Whisker.low	Econom
Country					

Brazil	22	6.635	6.725470	6.544531	1.107353
Mexico	25	6.578	6.671149	6.484851	1.153184
Thailand	32	6.424	6.509117	6.338883	1.127869
Colombia	36	6.357	6.452020	6.261980	1.070622
Malaysia	42	6.084	6.179980	5.988021	1.291215
Ecuador	44	6.008	6.105848	5.910152	1.000820
Poland	46	5.973	6.053908	5.892092	1.291788
Algeria	53	5.872	5.978286	5.765714	1.091864
Turkmenistan	59	5.822	5.885181	5.758819	1.130777
Kazakhstan	60	5.819	5.903642	5.734358	1.284556
Peru	63	5.715	5.811947	5.618054	1.035225
Libya	68	5.525	5.676954	5.373046	1.101803
Turkey	69	5.500	5.594865	5.405135	1.198274
Paraguay	70	5.493	5.577381	5.408619	0.932537
Serbia	73	5.395	5.491570	5.298430	1.069318
Jordan	74	5.336	5.448410	5.223590	0.991012
Hungary	75	5.324	5.403040	5.244960	1.286012
Jamaica	76	5.311	5.581399	5.040601	0.925579
China	79	5.273	5.319278	5.226721	1.081166
Montenegro	83	5.237	5.341044	5.132956	1.121129
Azerbaijan	85	5.234	5.299287	5.168714	1.153602
Dominican Republic	86	5.230	5.349061	5.110939	1.079374
Lebanon	88	5.225	5.318882	5.131118	1.074988
South Africa	101	4.829	4.929435	4.728565	1.054699
Bulgaria	105	4.714	4.803695	4.624306	1.161459
Iraq	117	4.497	4.622591	4.371409	1.102710
Gabon	118	4.465	4.557362	4.372639	1.198210
Angola	140	3.795	3.951642	3.638358	0.858428

In [72]:

```
f,ax = plt.subplots(figsize=(25, 18))
sns.heatmap(upperMiddleincomea_h.corr(), annot=True, linewidths=.5, fmt= '.1f',a
x=ax)
plt.show()
#note that the only important line in the heat map above is the last line.
```



Found Negative Correlation

Found Small Negative Correlation in Lower Middle Income Countries

High-Income Countries (GDP per capita > 12,615)

In [73]:

```
high_income = un_report[un_report["GDP per capita (current US$)"] > 12615]
hi_gdpc = high_income["GDP per capita (current US$)"]
```

In [74]:

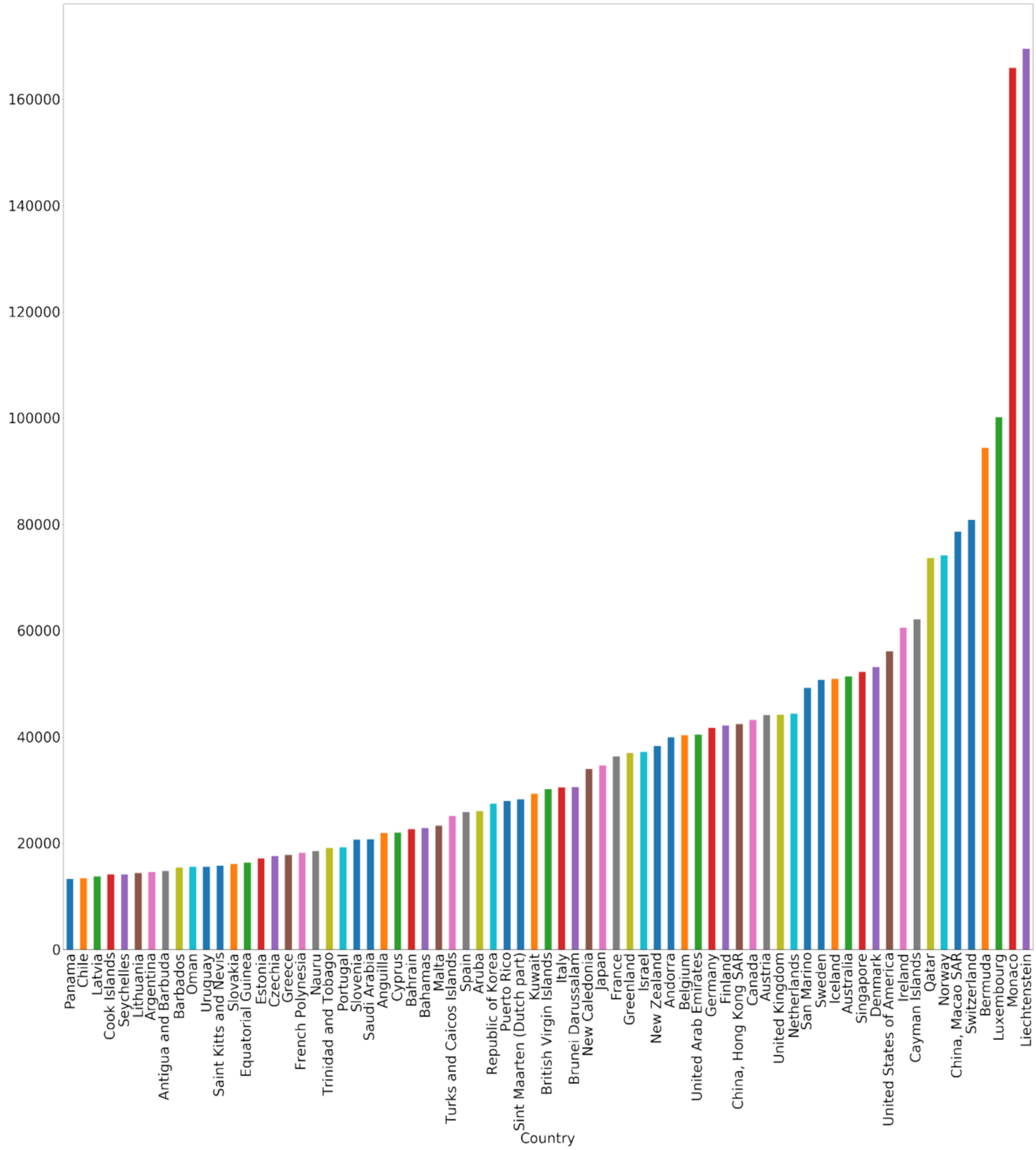
```
hi_gdpc = hi_gdpc.sort_values()
```

In [75]:

```
matplotlib.rcParams.update({'font.size': 40})
fig = plt.figure()
fig.set_size_inches(50,50)
hi_gdpc.plot("bar")
```

Out[75]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0b8b6b70>
```



In [76]:

```
hi_gdpc.describe()
```

Out[76]:

```
count        71.000000
mean      38491.988732
std       29897.125398
min       13268.100000
25%       18315.100000
50%       30144.500000
75%       44247.250000
max      169491.800000
Name: GDP per capita (current US$), dtype: float64
```

Happiness Scores of High-Income Income Countries

In [77]:

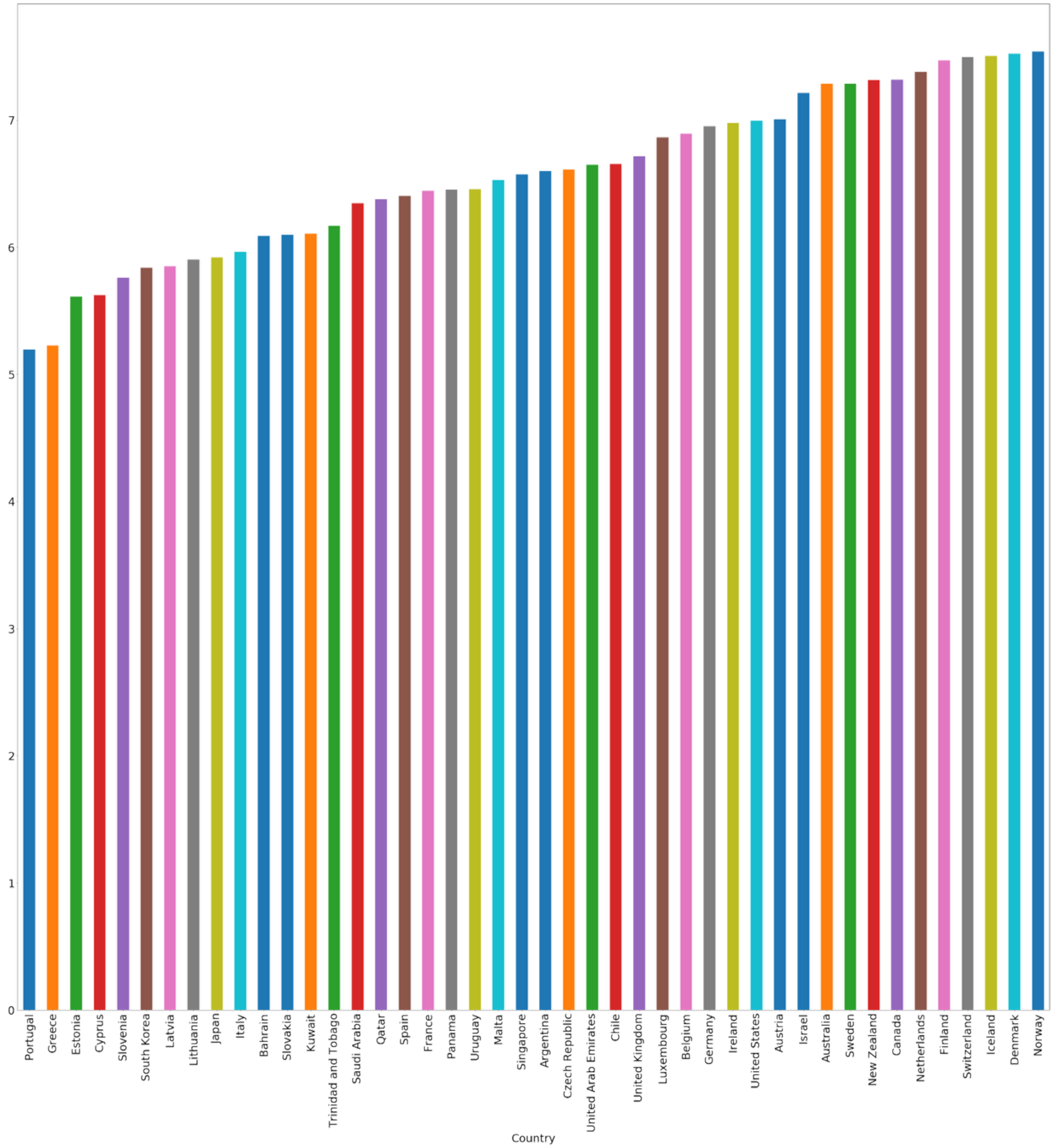
```
highIncome_hscore = h_score[hi_gdpc.index]
highIncome_hscore["United States"] = h_score["United States"]
highIncome_hscore["Czech Republic"] = h_score["Czech Republic"]
highIncome_hscore["South Korea"] = h_score["South Korea"]
highIncome_hscore = highIncome_hscore.dropna()
highIncome_hscore = highIncome_hscore.sort_values()
```

In [78]:

```
matplotlib.rcParams.update({'font.size': 30})
fig = plt.figure()
fig.set_size_inches(50,50)
highIncome_hscore.plot("bar")
```

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0af0df60>
```



In [79]:

```
highIncome_hscore.describe()
```

Out[79]:

```
count    44.000000
mean      6.571136
std       0.645832
min       5.195000
25%       6.095250
50%       6.585500
75%       7.057750
max       7.537000
Name: Happiness.Score, dtype: float64
```

Terrorist Attacks in High-Income Income Countries

In [80]:

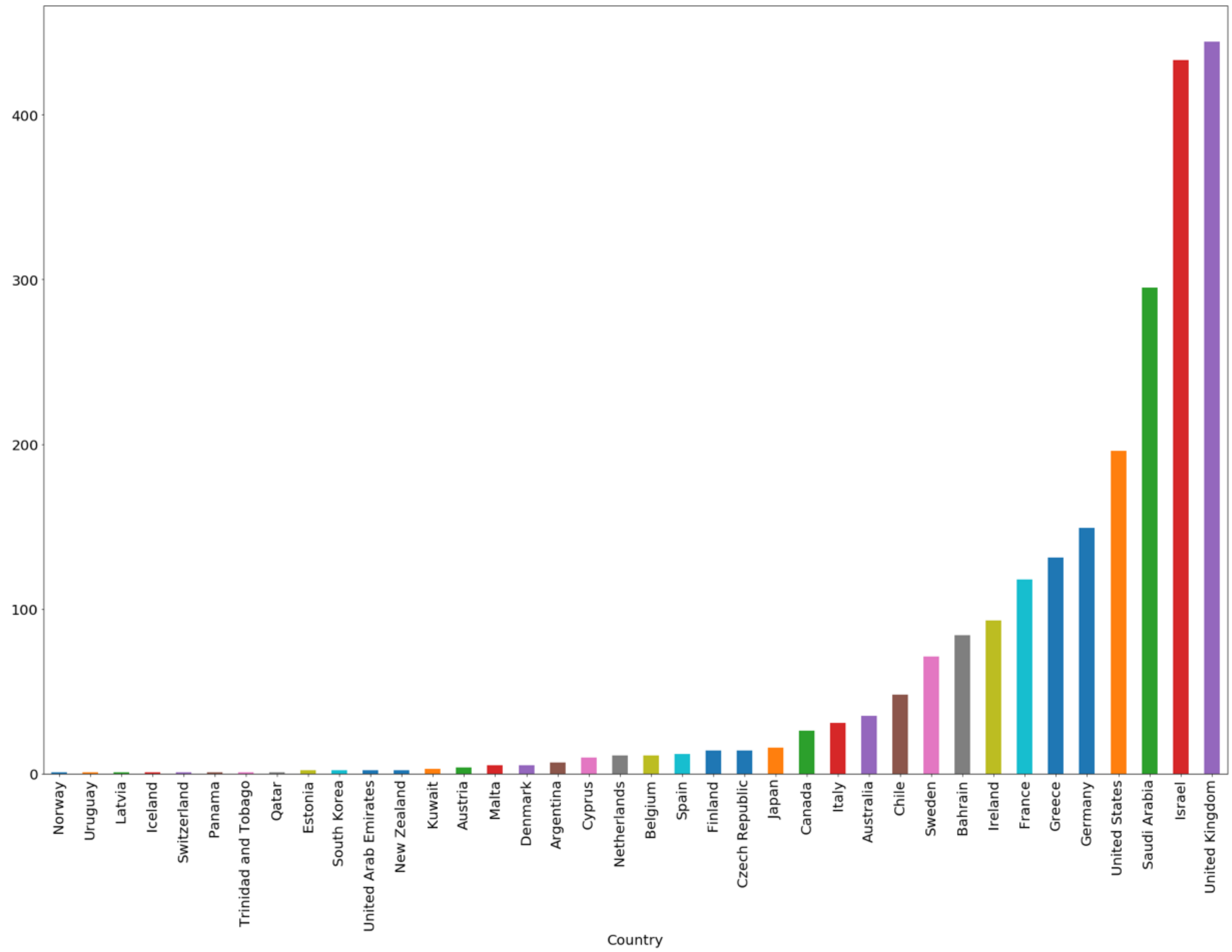
```
tattacks_hi = tattacks_count[highIncome_hscore.index]
tattacks_hi = tattacks_hi.dropna()
tattacks_hi = tattacks_hi.sort_values()
```

In [81]:

```
matplotlib.rcParams.update({'font.size': 20})
fig = plt.figure()
fig.set_size_inches(30,20)
tattacks_hi.plot("bar")
```

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0b6947f0>
```



In [82]:

```
tattacks_hi.describe()
```

Out[82]:

```
count     38.000000
mean      60.052632
std      110.710957
min        1.000000
25%        2.000000
50%       11.000000
75%       65.250000
max      444.000000
Name: Terrorist Attacks Count, dtype: float64
```

Relationship between the number Terrorism Attacks and Happiness Score of High-Income Countries

In [83]:

```
tattacks_hi.head()
```

Out[83]:

```
Country
Norway          1.0
Uruguay         1.0
Latvia          1.0
Iceland         1.0
Switzerland     1.0
Name: Terrorist Attacks Count, dtype: float64
```

In [84]:

```
highIncome_hscore.head()
```

Out[84]:

```
Country
Portugal    5.195
Greece      5.227
Estonia     5.611
Cyprus      5.621
Slovenia    5.758
Name: Happiness.Score, dtype: float64
```

In [85]:

```
tattacks_hi.corr(highIncome_hscore)
```

Out[85]:

```
0.048959696259354 82
```

In [86]:

```
Highincomea_h = happiness.copy()
Highincomea_h["Attacks"] = tattacks_hi
Highincomea_h = Highincomea_h.dropna()
```

In [ ]:

```
f,ax = plt.subplots(figsize=(25, 18))
sns.heatmap(Highincomea_h.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```

In [ ]:

```
tattacks_hi.sum()
tattacks_umi.sum()
tattacks_lmi.sum()
tattacks_lowincome.sum()
```

In [ ]:

```
classes = pd.Series(index=["Low Income","Lower Middle Income","Upper Middle Inco
me", "High Income" ])
classes.index.name="Classification"
classes.name="Total Terrorist Attacks"
classes["Low Income"] = tattacks_lowincome.sum()
classes["Lower Middle Income"] = tattacks_lmi.sum()
classes["Upper Middle Income"] = tattacks_umi.sum() - tattacks_count["Iraq"] #le
ave or remove Iraq???
classes["High Income"] = tattacks_hi.sum()
classes
```

In [ ]:

```
classes.plot("bar")
```

High Income Countries Stats

Happiness from 5.195(Portugal) to 7.537(Norway)

Mean Happiness = 6.57

Standard Deviation = [illegible]

In [ ]:

```
highIncome_hscore.mean()
```

In [ ]:

```
highIncome_hscore.std()
```

In [ ]:

```
from bokeh.io import output_file, output_notebook, show
from bokeh.io import output_file, show
from bokeh.models import ColumnDataSource, GMapOptions
from bokeh.plotting import gmap
```

Visualtization: Google Maps API

Shows terrorism attacks across the world.

In [ ]:

```
df = pd.read_csv('globalTerrorism.csv', engine = 'python')
```

In [ ]:

```
latitude_list = df['latitude'].tolist()
longitude_list = df['longitude'].tolist()
```

In [ ]:

```
map_options = GMapOptions(lat=29, lng=-98, map_type="roadmap", zoom=7)

p = gmap("AIzaSyAx7GNIhrPKKX7LPlC84iGc3Mg-8_uKdbI", map_options, title="Terroris
t Attacks")

source = ColumnDataSource(
    data=dict(lat=df['latitude'].tolist(),
              lon=df['longitude'].tolist())
)

p.circle(x="lon", y="lat", size=15, fill_color="blue", fill_alpha=0.8, source=so
urce)
output_notebook()
show(p)
```