In [7]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [8]:

```
#matplotlib setting
mpl.rcParams['figure.dpi']=200
mpl.rcParams['axes.spines.top']=False
mpl.rcParams['axes.spines.right']=False
```

In [9]:

```
df=pd.read_csv("world-happiness-report-2021.csv")
```

In [10]:

```
df
```

Out[10]:

	Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perce corr
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	72.000	0.949	-0.098	
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	72.700	0.946	0.030	
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	74.400	0.919	0.025	
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	73.000	0.955	0.160	
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	72.400	0.913	0.175	
...	...	...	...	...	...	...	...	...	...	...	...	...
144	Lesotho	Sub-Saharan Africa	3.512	0.120	3.748	3.276	7.926	0.787	48.700	0.715	-0.131	
145	Botswana	Sub-Saharan Africa	3.467	0.074	3.611	3.322	9.782	0.784	59.269	0.824	-0.246	
146	Rwanda	Sub-Saharan Africa	3.415	0.068	3.548	3.282	7.676	0.552	61.400	0.897	0.061	
147	Zimbabwe	Sub-Saharan Africa	3.145	0.058	3.259	3.030	7.943	0.750	56.201	0.677	-0.047	
148	Afghanistan	South Asia	2.523	0.038	2.596	2.449	7.695	0.463	52.493	0.382	-0.102	

149 rows × 20 columns

In [11]:

```
df.head()
```

Out[11]:

	Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Percepti corrup
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	72.0	0.949	-0.098	0.
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	72.7	0.946	0.030	0.
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	74.4	0.919	0.025	0.
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	73.0	0.955	0.160	0.
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	72.4	0.913	0.175	0.

In [12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 20 columns):
 #   Column                                    Non-Null Count  Dtype  
---  ------                                    --------------  -----  
 0   Country name                              149 non-null    object 
 1   Regional indicator                        149 non-null    object 
 2   Ladder score                              149 non-null    float64
 3   Standard error of ladder score            149 non-null    float64
 4   upperwhisker                              149 non-null    float64
 5   lowerwhisker                              149 non-null    float64
 6   Logged GDP per capita                     149 non-null    float64
 7   Social support                            149 non-null    float64
 8   Healthy life expectancy                   149 non-null    float64
 9   Freedom to make life choices              149 non-null    float64
 10  Generosity                                149 non-null    float64
 11  Perceptions of corruption                 149 non-null    float64
 12  Ladder score in Dystopia                  149 non-null    float64
 13  Explained by: Log GDP per capita          149 non-null    float64
 14  Explained by: Social support              149 non-null    float64
 15  Explained by: Healthy life expectancy     149 non-null    float64
 16  Explained by: Freedom to make life choices  149 non-null  float64
 17  Explained by: Generosity                  149 non-null    float64
 18  Explained by: Perceptions of corruption   149 non-null    float64
 19  Dystopia + residual                       149 non-null    float64
dtypes: float64(18), object(2)
memory usage: 23.4+ KB
```

In [13]:

```
df.describe()
```

Out[13]:

	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
count	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000
mean	5.532839	0.058752	5.648007	5.417631	9.432208	0.814745	64.992799	0.791597	-0.015134	0.727450
std	1.073924	0.022001	1.054330	1.094879	1.158601	0.114889	6.762043	0.113332	0.150657	0.179226
min	2.523000	0.026000	2.596000	2.449000	6.635000	0.463000	48.478000	0.382000	-0.288000	0.082000
25%	4.852000	0.043000	4.991000	4.706000	8.541000	0.750000	59.802000	0.718000	-0.126000	0.667000
50%	5.534000	0.054000	5.625000	5.413000	9.569000	0.832000	66.603000	0.804000	-0.036000	0.781000
75%	6.255000	0.070000	6.344000	6.128000	10.421000	0.905000	69.600000	0.877000	0.079000	0.845000

max	7.842000	0.173000	7.904000	7.780000	11.647000	0.983000	76.953000	0.970000	0.542000	0.939000

Standard

Freedom

In [14]:

```
df.describe().T.style.bar(subset=['mean'], color='#205ff2')\
                            .background_gradient(subset=['std'], cmap='Reds')\
                              .background_gradient(subset=['50%'], cmap='coolwarm')
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
Ladder score	149.000000	5.532839	1.073924	2.523000	4.852000	5.534000	6.255000	7.842000
Standard error of ladder score	149.000000	0.058752	0.022001	0.026000	0.043000	0.054000	0.070000	0.173000
upperwhisker	149.000000	5.648007	1.054330	2.596000	4.991000	5.625000	6.344000	7.904000
lowerwhisker	149.000000	5.417631	1.094879	2.449000	4.706000	5.413000	6.128000	7.780000
Logged GDP per capita	149.000000	9.432208	1.158601	6.635000	8.541000	9.569000	10.421000	11.647000
Social support	149.000000	0.814745	0.114889	0.463000	0.750000	0.832000	0.905000	0.983000
Healthy life expectancy	149.000000	64.992799	6.762043	48.478000	59.802000	66.603000	69.600000	76.953000
Freedom to make life choices	149.000000	0.791597	0.113332	0.382000	0.718000	0.804000	0.877000	0.970000
Generosity	149.000000	-0.015134	0.150657	-0.288000	-0.126000	-0.036000	0.079000	0.542000
Perceptions of corruption	149.000000	0.727450	0.179226	0.082000	0.667000	0.781000	0.845000	0.939000
Ladder score in Dystopia	149.000000	2.430000	0.000000	2.430000	2.430000	2.430000	2.430000	2.430000
Explained by: Log GDP per capita	149.000000	0.977161	0.404740	0.000000	0.666000	1.025000	1.323000	1.751000
Explained by: Social support	149.000000	0.793315	0.258871	0.000000	0.647000	0.832000	0.996000	1.172000
Explained by: Healthy life expectancy	149.000000	0.520161	0.213019	0.000000	0.357000	0.571000	0.665000	0.897000
Explained by: Freedom to make life choices	149.000000	0.498711	0.137888	0.000000	0.409000	0.514000	0.603000	0.716000
Explained by: Generosity	149.000000	0.178047	0.098270	0.000000	0.105000	0.164000	0.239000	0.541000
Explained by: Perceptions of corruption	149.000000	0.135141	0.114361	0.000000	0.060000	0.101000	0.174000	0.547000
Dystopia + residual	149.000000	2.430329	0.537645	0.648000	2.138000	2.509000	2.794000	3.482000

In [15]:

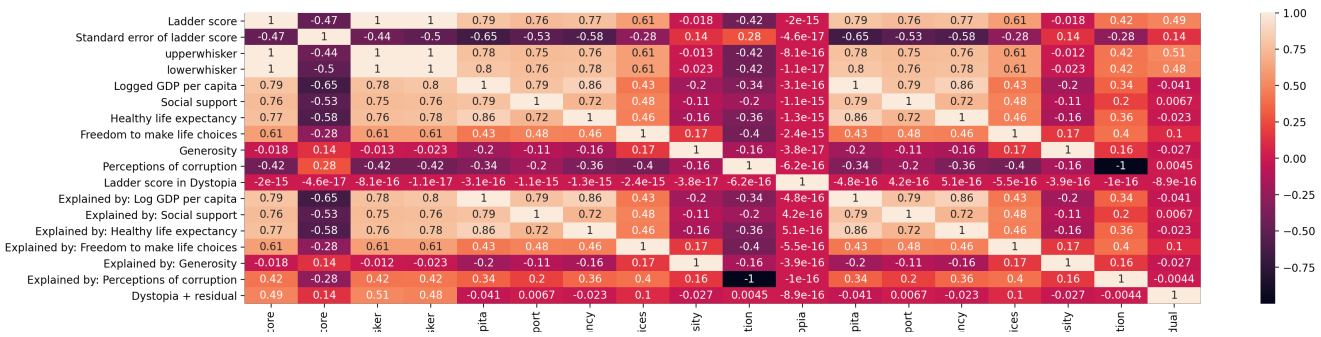
```
#Above data is the statistical information of the dataframe

#Bright red is the standard deviation value 6.7620 because it is the highest amon standard deviati
ons! Healthy Life expectancy have highest standard deviation
```

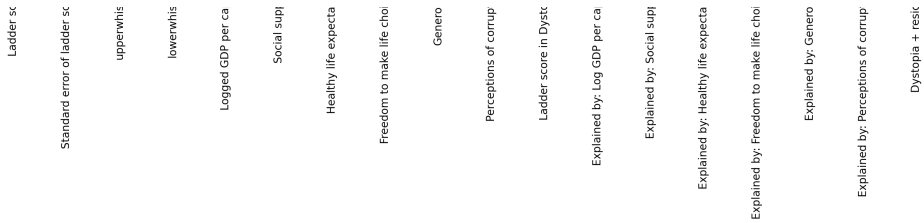
In [16]:

```
plt.figure(figsize=(20,5))
sns.heatmap(df.corr(),annot=True)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24069dca370>
```





In [17]:

```
high_corruption=df[['Country name','Perceptions of corruption','Healthy life expectancy']].sort_va
lues('Perceptions of corruption', ascending=False)
```

In [18]:

```
import plotly.express as px
```

In [19]:

```
fig=px.bar(high_corruption[:30],x='Country name', y='Perceptions of corruption',color='Healthy
life expectancy', title='High corruption countries')
fig.show()
```

In [20]:

```
#Top 30 countries with highest corruption

Afghanistan, Lesotho, Nigeria, Sierra Leone has the least healthy life expectancy.
```

```
  File "<ipython-input-20-461d14cc0f83>", line 3
    Afghanistan, Lesotho, Nigeria, Sierra Leone has the least healthy life expectancy.
                                   ^
SyntaxError: invalid syntax
```

In [21]:

```
##Let's see the top countries with least corruption
```

```
fig=px.bar(high_corruption[120:],x='Country name',y='Perceptions of corruption',color='Healthy life
expectancy',title='High corruption countries')
fig.show()
```

In [22]:

```
life_exp=df[['Country name','Freedom to make life choices','Healthy life expectancy']].sort_values
('Healthy life expectancy',ascending=False)
```

In [24]:

```
fig=px.bar(life_exp[:20],x='Country name',y='Healthy life expectancy',color='Freedom to make life
choices',title='Low life expentancy and freedom for life choices')
fig.show()
```

In [ ]:

```
#Singapore has the highest healthy life expectancy with value of 76.953 and freedom to make life c
hoice is also very high .927
```

In [25]:

```
fig = px.bar(life_exp[140:], x='Country name', y='Healthy life expectancy', color ='Freedom to make
life choices',
             title = 'Low life expectancy and freedom for life choices')
fig.show()
```

In [ ]:

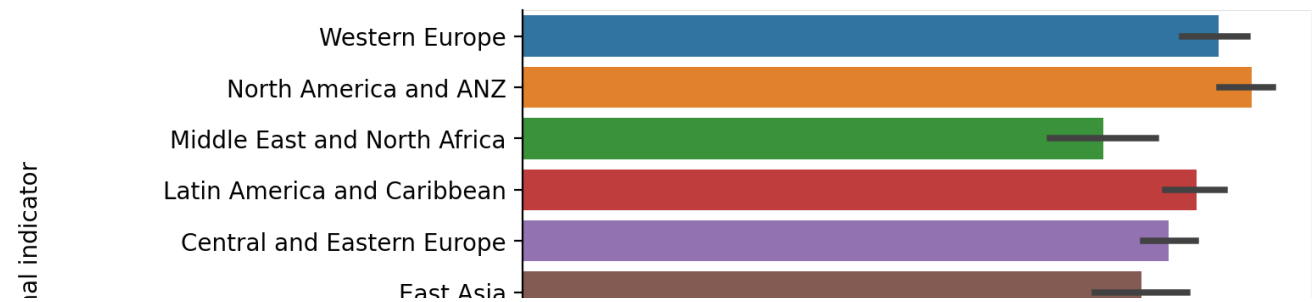
```
#Chad, Lesotho, Nigeria have least healthy life expectancy
```

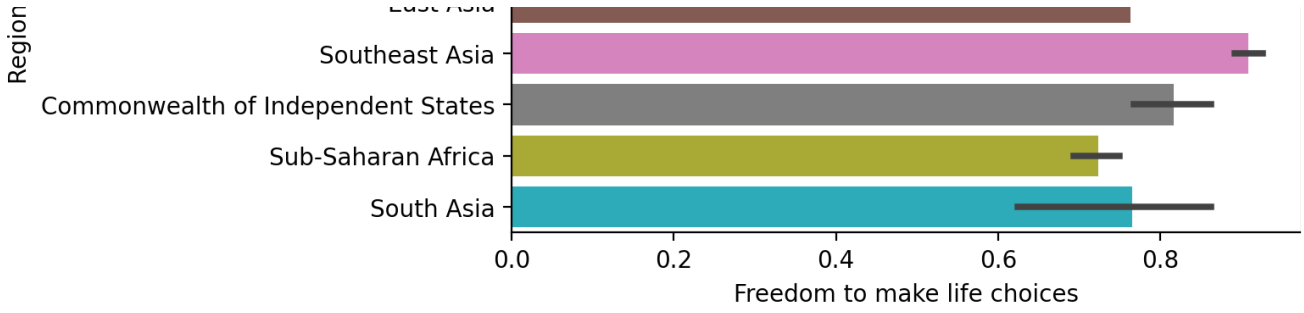
In [27]:

```
sns.barplot(y='Regional indicator',x='Freedom to make life choices',data=df)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2406af88460>
```





In [29]:

```
fig=px.scatter(df,x='Regional indicator',y='Freedom to make life choices',color="Healthy life expectancy",size='Logged GDP per capita',hover_data=['Social support'])
fig.show()
```

In [31]:

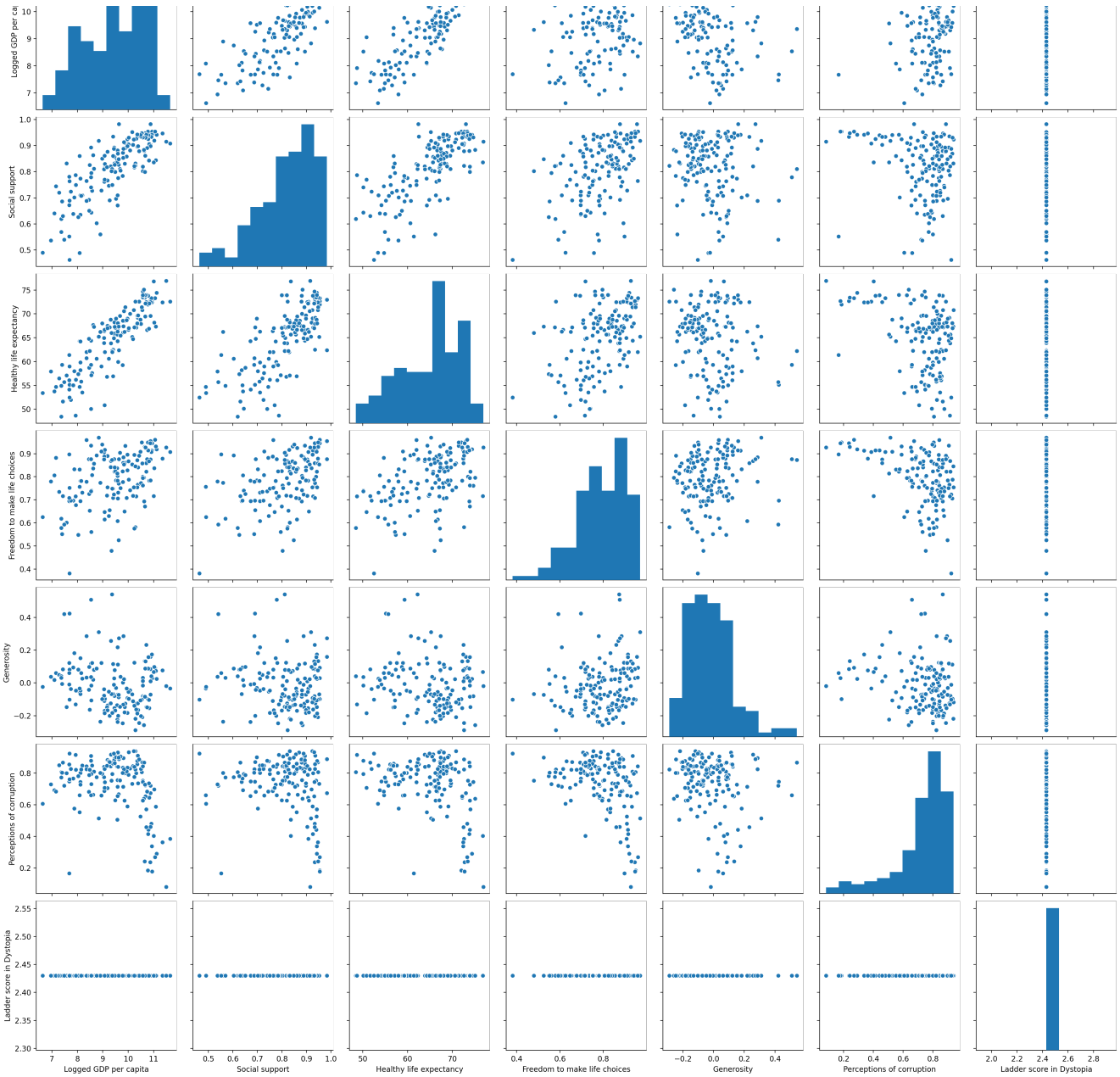
```
#Move your cursor around the bubbles to get detailed information

#For example if you put your cursor on the lowest bubble of South Asia you will get information like Freedom to make life choices = 0.382, Logged GDP per capita 7.695, Social support 0.463, Healthy life expectancy =52.493
```

In [32]:

```
cols = ['Logged GDP per capita','Social support', 'Healthy life expectancy', 'Freedom to make life choices',
        'Generosity', 'Perceptions of corruption', 'Ladder score in Dystopia']

sns.pairplot(df[cols], height = 3)
```

Out[32]:

```
<seaborn.axisgrid.PairGrid at 0x2406b404c70>
```



In [33]:

```
#Logged GDP per capita has strong correlation with Healthy Life expectancy and Social support, som
ewhat good correlation with Freedom to make life choices
#Social Support has strong correlation with Healthy Life expectancy and somewhat good correlation
with Freedom to make life choices
#Healthy Life expectancy has strong correlation with Logged GDP per capita and Social support, som
ewhat strong correlation with Freedom to make life choices
#Freedom to make life choices has somewhat strong correlation with Logged GDP per capita, Healthy
Life expectancy, Social support and Generosity
```

In [34]:

```
fig = px.pie(df, values='Logged GDP per capita', names='Regional indicator', title='% of Logged GDP
of regions from data')
fig.show()
```

In [35]:

```
#20.7% of data has Sub Saharan African region.

#Log of real GDP per capita Natural logs have a few great properties for our purposes. Using them
means that every step up the y-axis is an identical percent change in real GDP per capita. Going f
rom 7.0 to 7.5, for example, is a 65% increase in real GDP per capita. Going from 7.5 to 8.0 is al
so a 65% increase in real GDP per capita.
```

In [36]:

```
fig = px.density_heatmap(df, x="Freedom to make life choices", y='Perceptions of corruption', margi
nal_x="box", marginal_y="violin")
fig.show()
```

In [37]:

```
#Freedom to make life choices has median value .804
#Perceptions of corruption has median value .781
#The square boxes gives the estimate of freedom to make life choices and perceptions of corruption
```

```
#The square boxes gives the estimate of freedom to make life choices and perceptions of corruption
s with counts, again the reader is advised to move around the cursor to understand the values
```

In [38]:

```
fig = px.scatter(df, x="Healthy life expectancy", y="Logged GDP per capita", color="Regional
indicator",marginal_x="box")
fig.show()
```

In [39]:

```
#A simple scatter plot to understand the correlation between healthy life expectancy and logged GD
P per capita along with the regions they represnt
#One can notice the pink dots in the left and bottom represents sub saharan Africa with healthy li
fe expectancy of 53.4
```

In [40]:

```
fig = px.violin(df, y="Logged GDP per capita",x ="Generosity", color = 'Regional indicator', box=Tr
ue, # draw box plot inside the violin
                points='all', hover_data=df.columns # can be 'outliers', or False
               )
fig.show()
```

In [41]:

```
#This maybe the most essential plot
#move your cursor around the points and you will get all detailed for each point(includes all the
informations like country name, logged GDP, generosity, etc all the columns)
```

In [43]:

```
sns.set_style("darkgrid")
```

In [44]:

```
#Ladder Score :
#The rankings of national happiness are based on a Cantril ladder survey. Nationally
representative samples of respondents are asked to think of a ladder, with the best possible life
for them being a 10, and the worst possible life being a 0. They are then asked to rate their own
current lives on that 0 to 10 scale.
```

In [45]:

```
ladder_score = df[['Ladder score',"Standard error of ladder score",'Country name']].sort_values("L
adder score",ascending=False)
```

In [46]:

```
ladder_score[:10]
```

Out[46]:

	Ladder score	Standard error of ladder score	Country name
0	7.842	0.032	Finland
1	7.620	0.035	Denmark
2	7.571	0.036	Switzerland
3	7.554	0.059	Iceland
4	7.464	0.027	Netherlands
5	7.392	0.035	Norway
6	7.363	0.036	Sweden
7	7.324	0.037	Luxembourg
8	7.277	0.040	New Zealand
9	7.268	0.036	Austria

In [47]:

```
# Seeing top 12 and bottom 12

fig = px.histogram(ladder_score.head(12), x = "Ladder score" , y = "Country name",
                   color='Country name',title="Top 12 Ladder Scores."); fig.show()

fig = px.histogram(ladder_score.tail(12), x = "Ladder score" , y = "Country name",
```

```
                    color='Country name', title="Bottom 12 Ladder scores"); fig.show()
```

In [ ]:

```
#Findand has the highest and Afghanistan has the lowest score.
```

In [48]:

```
#Logged GDP per capita
gdp = df[["Logged GDP per capita", "Country name","Regional indicator"]].sort_values("Logged GDP p
```

```
er capita",ascending=False)
# Top 10
gdp.head(10)
```

Out[48]:

	Logged GDP per capita	Country name	Regional indicator
7	11.647	Luxembourg	Western Europe
31	11.488	Singapore	Southeast Asia
14	11.342	Ireland	Western Europe
2	11.117	Switzerland	Western Europe
24	11.085	United Arab Emirates	Middle East and North Africa
5	11.053	Norway	Western Europe
18	11.023	United States	North America and ANZ
76	11.000	Hong Kong S.A.R. of China	East Asia
1	10.933	Denmark	Western Europe
4	10.932	Netherlands	Western Europe

In [49]:

```
# Botttom 10
gdp.tail(10)
```

Out[49]:

	Logged GDP per capita	Country name	Regional indicator
142	7.477	Haiti	Latin America and Caribbean
137	7.434	Sierra Leone	Sub-Saharan Africa
134	7.396	Madagascar	Sub-Saharan Africa
127	7.364	Chad	Sub-Saharan Africa
135	7.362	Togo	Sub-Saharan Africa
119	7.288	Liberia	Sub-Saharan Africa
114	7.158	Mozambique	Sub-Saharan Africa
95	7.098	Niger	Sub-Saharan Africa
143	6.958	Malawi	Sub-Saharan Africa
139	6.635	Burundi	Sub-Saharan Africa

In [50]:

```
print("Average GDP : {}".format(np.mean(gdp['Logged GDP per capita'])))
```

```
Average GDP : 9.432208053691266
```

In [51]:

```
fig = px.scatter(gdp, x = 'Logged GDP per capita', y = 'Logged GDP per capita', size='Logged GDP pe
r capita',
                 color = "Country name",hover_data=["Regional indicator"],title="GDP")
fig.show()
```

In [52]:

```
#Luxembourg has the highes GDP per capita at 11.647.
```

In [53]:

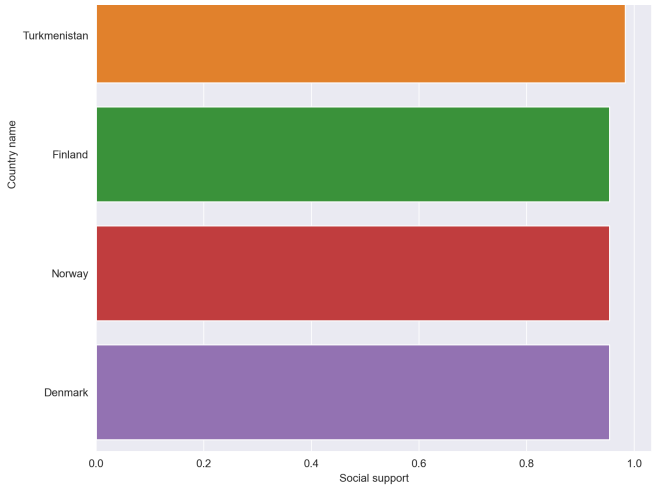
```
#Social Support
social_support = df[["Social support","Country name"]].sort_values("Social support",ascending = False)
social_support.tail(10)
```

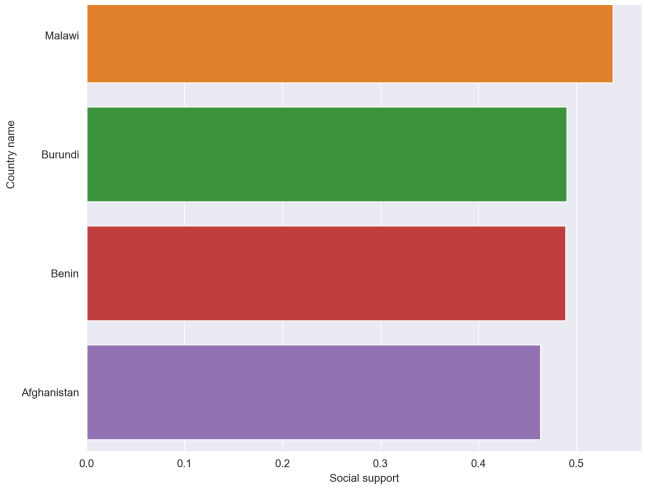
Out[53]:

	Social support	Country name
127	0.619	Chad
138	0.603	India
135	0.569	Togo
105	0.560	Morocco
146	0.552	Rwanda
142	0.540	Haiti
143	0.537	Malawi
139	0.490	Burundi
98	0.489	Benin
148	0.463	Afghanistan

In [54]:

```
plt.figure(figsize = [20,10])

plt.subplot(121); sns.barplot(data = social_support.head(5),x='Social support',y = "Country name");
plt.title("Top 5")
plt.subplot(122); sns.barplot(data = social_support.tail(5),x='Social support',y = "Country name");
plt.title("Bottom 5");
```





In [55]:

```
#Iceland again at top and Afghanistan again at bottom.
```

In [56]:

```
#Healthy life expectancy & Freedom to make life choices
hlef = df[["Freedom to make life choices","Healthy life expectancy","Country name","Regional indic
ator"]].sort_values("Healthy life expectancy",ascending=False)
hlef.head()
```

Out[56]:

	Freedom to make life choices	Healthy life expectancy	Country name	Regional indicator
31	0.927	76.953	Singapore	Southeast Asia
76	0.717	76.820	Hong Kong S.A.R. of China	East Asia
55	0.796	75.100	Japan	East Asia
26	0.761	74.700	Spain	Western Europe
2	0.919	74.400	Switzerland	Western Europe

In [57]:

```
fig = px.scatter(hlef, x = "Healthy life expectancy", y = "Freedom to make life choices",
                 size = "Freedom to make life choices", color = "Regional indicator",
                 hover_name="Country name")
fig.show()
```

In [59]:

```
#Sub-Saharan Africa seems to have very low healty life expectancy.
```

In [60]:

```
#Generosity and Perceptions of corruption
gen_cor = df[["Perceptions of corruption","Generosity","Country name",
              "Regional indicator","Freedom to make life choices"]].sort_values("Perceptions of cor
ruption",ascending=False)

# Top corrupt
gen_cor.head(20)
```

Out[60]:

	Perceptions of corruption	Generosity	Country name	Regional indicator	Freedom to make life choices
59	0.939	-0.118	Croatia	Central and Eastern Europe	0.754
45	0.938	-0.219	Romania	Central and Eastern Europe	0.845
87	0.932	-0.096	Bulgaria	Central and Eastern Europe	0.788
63	0.931	0.113	Bosnia and Herzegovina	Central and Eastern Europe	0.706
148	0.924	-0.102	Afghanistan	South Asia	0.382
109	0.924	-0.011	Ukraine	Commonwealth of Independent States	0.724
64	0.918	-0.079	Moldova	Commonwealth of Independent States	0.822
32	0.917	0.257	Kosovo	Central and Eastern Europe	0.869
144	0.915	-0.131	Lesotho	Sub-Saharan Africa	0.715
33	0.911	-0.124	Slovakia	Central and Eastern Europe	0.766
66	0.908	0.119	Kyrgyzstan	Commonwealth of Independent States	0.935
93	0.905	0.038	North Macedonia	Central and Eastern Europe	0.751
92	0.901	-0.030	Albania	Central and Eastern Europe	0.785
122	0.898	-0.073	Lebanon	Middle East and North Africa	0.525
53	0.895	0.287	Thailand	Southeast Asia	0.884
62	0.891	-0.154	Peru	Latin America and Caribbean	0.822
96	0.888	0.273	Turkmenistan	Commonwealth of Independent States	0.877
57	0.887	-0.244	Portugal	Western Europe	0.892
36	0.884	-0.137	Jamaica	Latin America and Caribbean	0.890
70	0.882	0.028	Paraguay	Latin America and Caribbean	0.876

In [61]:

```
# Least corrupt
gen_cor.tail(20)
```

Out[61]:

	Perceptions of corruption	Generosity	Country name	Regional indicator	Freedom to make life choices
39	0.527	-0.106	Estonia	Central and Eastern Europe	0.909

	Perceptions of corruption	Generosity	Country name	Regional indicator	Freedom to make life choices
41	0.515	0.311	Uzbekistan	Commonwealth of Independent States	0.970
89	0.500	-0.223	Azerbaijan	Commonwealth of Independent States	0.814
9	0.481	0.042	Austria	Western Europe	0.908
12	0.460	0.011	Germany	Western Europe	0.875
16	0.459	0.233	United Kingdom	Western Europe	0.859
10	0.442	0.159	Australia	North America and ANZ	0.914
13	0.415	0.089	Canada	North America and ANZ	0.915
76	0.403	0.067	Hong Kong S.A.R. of China	East Asia	0.717
7	0.386	-0.034	Luxembourg	Western Europe	0.907
14	0.363	0.077	Ireland	Western Europe	0.879
4	0.338	0.175	Netherlands	Western Europe	0.913
2	0.292	0.025	Switzerland	Western Europe	0.919
5	0.270	0.093	Norway	Western Europe	0.960
8	0.242	0.134	New Zealand	North America and ANZ	0.929
6	0.237	0.086	Sweden	Western Europe	0.945
0	0.186	-0.098	Finland	Western Europe	0.949
1	0.179	0.030	Denmark	Western Europe	0.946
146	0.167	0.061	Rwanda	Sub-Saharan Africa	0.897
31	0.082	-0.018	Singapore	Southeast Asia	0.927

In [62]:

```
fig = px.scatter_3d(gen_cor,x = "Generosity", y = "Freedom to make life choices" , z = "Perceptions of corruption",
                    color="Regional indicator",hover_name="Country name",height=800,width=1000)
fig.show()
```

In [63]:

```
#There are a lot of corrupt countries compared with less corrupt ones.
#Central and Eastern Europe seems to be mostly corrupt.
#Most countries seem to fall on the negative side in terms of Generosity.
```

In [66]:

```
import statsmodels.formula.api as smf

from scipy.stats import stats
from scipy import stats
import pylab #ggplot

from statsmodels.stats.multicomp import (pairwise_tukeyhsd,MultiComparison) #Tukey

import warnings
warnings.filterwarnings('ignore')
```

In [71]:

```
#Descriptive Statistics
summary = df.describe().T
se = df.describe().T['std']/np.sqrt(df.describe().T['count'])
summary.insert(3,'se',se)
summary['Skewness'] = stats.skew(df._get_numeric_df(), nan_policy='omit')
summary['Kurtosis'] = stats.kurtosis(df._get_numeric_df(), nan_policy='omit')
summary
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-71-530aeb5b8403> in <module>
      3 se = df.describe().T['std']/np.sqrt(df.describe().T['count'])
      4 summary.insert(3,'se',se)
----> 5 summary['Skewness'] = stats.skew(df._get_numeric_df(), nan_policy='omit')
      6 summary['Kurtosis'] = stats.kurtosis(df._get_numeric_df(), nan_policy='omit')
      7 summary

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5272             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   5273                 return self[name]
-> 5274             return object.__getattribute__(self, name)
   5275
   5276     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute '_get_numeric_df'
```

In [ ]:

```

```

In [ ]:

```

```