In [49]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\christopher bradway\anac
onda3\lib\site-packages (4.12.0)
Requirement already satisfied: six in c:\users\christopher bradway\anacond
a3\lib\site-packages (from plotly) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\christopher bra
dway\anaconda3\lib\site-packages (from plotly) (1.3.3)
```

In [50]:
```
!pip install chart_studio
```

```
Requirement already satisfied: chart_studio in c:\users\christopher bradwa
y\anaconda3\lib\site-packages (1.1.0)
Requirement already satisfied: plotly in c:\users\christopher bradway\anac
onda3\lib\site-packages (from chart_studio) (4.12.0)
Requirement already satisfied: requests in c:\users\christopher bradway\an
aconda3\lib\site-packages (from chart_studio) (2.24.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\christopher bra
dway\anaconda3\lib\site-packages (from chart_studio) (1.3.3)
Requirement already satisfied: six in c:\users\christopher bradway\anacond
a3\lib\site-packages (from chart_studio) (1.15.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\christopher bradwa
y\anaconda3\lib\site-packages (from requests->chart_studio) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\christopher
bradway\anaconda3\lib\site-packages (from requests->chart_studio) (2020.6.
20)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\christopher b
radway\anaconda3\lib\site-packages (from requests->chart_studio) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\christopher bradway\anaconda3\lib\site-packages (from requests->c
hart_studio) (1.25.9)
```

In [51]:
```
!pip install discover_feature_relationships
```

```
Requirement already satisfied: discover_feature_relationships in c:\users\
christopher bradway\anaconda3\lib\site-packages (1.0.3)
```

In [52]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import os
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import seaborn as sns
import plotly.graph_objs as pgo
import plotly.express as px
from plotly.offline import download_plotlyjs, init_notebook_mode, plot,ipl
ot
import chart_studio.plotly as csp
init_notebook_mode (connected=True)
import statsmodels.formula.api as stats
```

```
from statsmodels.formula.api import ols
from discover_feature_relationships import discover
from sklearn.model_selection import cross_val_score
```

In [53]:
```
df2015=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2015.csv')
df2015['Year'] = 2015
df2015.head()
```

Out[53]:

	Country	Region	Rank	Score	GDP	Support	Health	Freedom	Corruption	Generosity	Y
0	Switzerland	Western Europe	1	7.587	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2(
1	Iceland	Western Europe	2	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2(
2	Denmark	Western Europe	3	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2(
3	Norway	Western Europe	4	7.522	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2(
4	Canada	North America	5	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2(

In [54]:
```
df2016=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2016.csv')
df2017=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2017.csv')
df2018=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2018.csv')
df2019=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2019.csv')
df2020=pd.read_csv(r'C:\Users\Christopher Bradway\Downloads\2020.csv')
```

In [55]:
```
for col in df2016.columns:
    print(col)
```

```
Country
Region
Rank
Score
GDP
Support
Health
Freedom
Corruption
Generosity
```

In [56]:
```
for col in df2019.columns:
    print(col)
```

```
Rank
Country
Score
GDP
Support
Health
Freedom
```

```
Generosity
Corruption
```

In [57]:
```
df2020.head()
```

Out[57]:

	Country	Regional	Score	GDP	Support	Health	Freedom	Generosity	Corruption	R
0	Finland	Western Europe	7.8087	10.639267	0.954330	71.900825	0.949172	-0.059482	0.195445	
1	Denmark	Western Europe	7.6456	10.774001	0.955991	72.402504	0.951444	0.066202	0.168489	
2	Switzerland	Western Europe	7.5599	10.979933	0.942847	74.102448	0.921337	0.105911	0.303728	
3	Iceland	Western Europe	7.5045	10.772559	0.974670	73.000000	0.948892	0.246944	0.711710	
4	Norway	Western Europe	7.4880	11.087804	0.952487	73.200783	0.955750	0.134533	0.263218	

In [58]:
```
df2017.head()
```

Out[58]:

	Country	Rank	Score	GDP	Support	Health	Freedom	Generosity	Corruption
0	Norway	1	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964
1	Denmark	2	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770
2	Iceland	3	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527
3	Switzerland	4	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007
4	Finland	5	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612

In [59]:
```
df2015.describe()
```

Out[59]:

	Rank	Score	GDP	Support	Health	Freedom	Corruption	Generosi
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.0000
mean	79.493671	5.375734	0.846137	0.991046	0.630259	0.428615	0.143422	0.2372
std	45.754363	1.145010	0.403121	0.272369	0.247078	0.150693	0.120034	0.1266
min	1.000000	2.839000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	40.250000	4.526000	0.545808	0.856823	0.439185	0.328330	0.061675	0.1505
50%	79.500000	5.232500	0.910245	1.029510	0.696705	0.435515	0.107220	0.2161
75%	118.750000	6.243750	1.158448	1.214405	0.811013	0.549092	0.180255	0.3098
max	158.000000	7.587000	1.690420	1.402230	1.025250	0.669730	0.551910	0.7958

In [60]:
```
df2015.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Country     158 non-null    object 
 1   Region      158 non-null    object 
 2   Rank        158 non-null    int64  
 3   Score       158 non-null    float64
 4   GDP         158 non-null    float64
 5   Support     158 non-null    float64
 6   Health      158 non-null    float64
 7   Freedom     158 non-null    float64
 8   Corruption  158 non-null    float64
 9   Generosity  158 non-null    float64
 10  Year        158 non-null    int64  
dtypes: float64(7), int64(2), object(2)
memory usage: 13.7+ KB
```

In [61]:
```
df2016['Year']=2016
df2017['Year']=2017
df2018['Year']=2018
df2019['Year']=2019
df2020['Year']=2020
```

In [62]:
```
df2018.head()
```

Out[62]:

	Rank	Country	Score	GDP	Support	Health	Freedom	Generosity	Corruption	Year
0	1	Finland	7.632	1.305	1.592	0.874	0.681	0.202	0.393	2018
1	2	Norway	7.594	1.456	1.582	0.861	0.686	0.286	0.340	2018
2	3	Denmark	7.555	1.351	1.590	0.868	0.683	0.284	0.408	2018
3	4	Iceland	7.495	1.343	1.644	0.914	0.677	0.353	0.138	2018
4	5	Switzerland	7.487	1.420	1.549	0.927	0.660	0.256	0.357	2018

In [63]:
```
df2020.head()
```

Out[63]:

	Country	Regional	Score	GDP	Support	Health	Freedom	Generosity	Corruption	R
0	Finland	Western Europe	7.8087	10.639267	0.954330	71.900825	0.949172	-0.059482	0.195445	
1	Denmark	Western Europe	7.6456	10.774001	0.955991	72.402504	0.951444	0.066202	0.168489	
2	Switzerland	Western Europe	7.5599	10.979933	0.942847	74.102448	0.921337	0.105911	0.303728	
3	Iceland	Western Europe	7.5045	10.772559	0.974670	73.000000	0.948892	0.246944	0.711710	

4	Norway	Western Europe	7.4880	11.087804	0.952487	73.200783	0.955750	0.134533	0.263218

In [64]:
```
df2015.shape
```

Out[64]: (158, 11)

In [65]:
```
df2016.shape
```

Out[65]: (157, 11)

In [66]:
```
df2017.shape
```

Out[66]: (155, 10)

In [67]:
```
df2018.shape
```

Out[67]: (156, 10)

In [68]:
```
df2019.shape
```

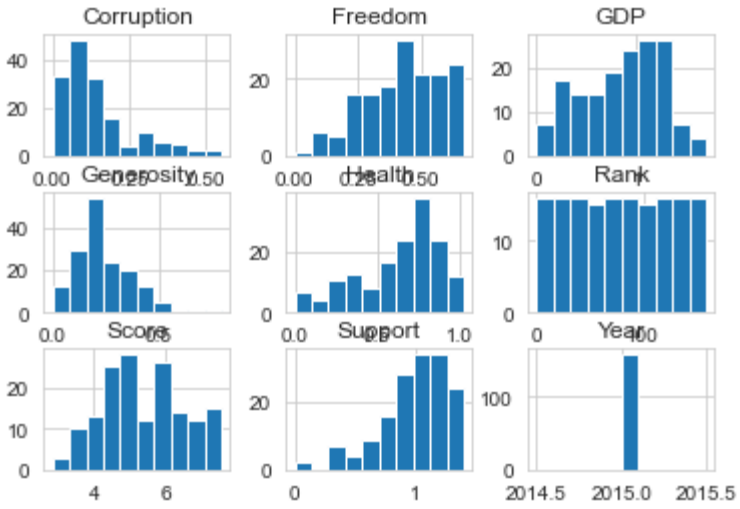
Out[68]: (156, 10)

In [69]:
```
df2020.shape
```

Out[69]: (153, 11)

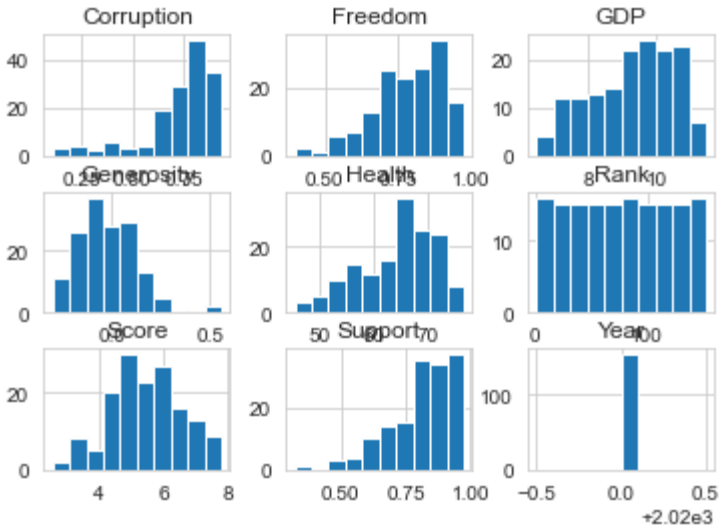
In [70]:
```
df2015.hist()
```

Out[70]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A5B494
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A7B4EE
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CBA44C
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A1CC91
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A19DD6
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CB8D13
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CB8D22
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B0556D
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8AE97EE
0>]],
      dtype=object)
```



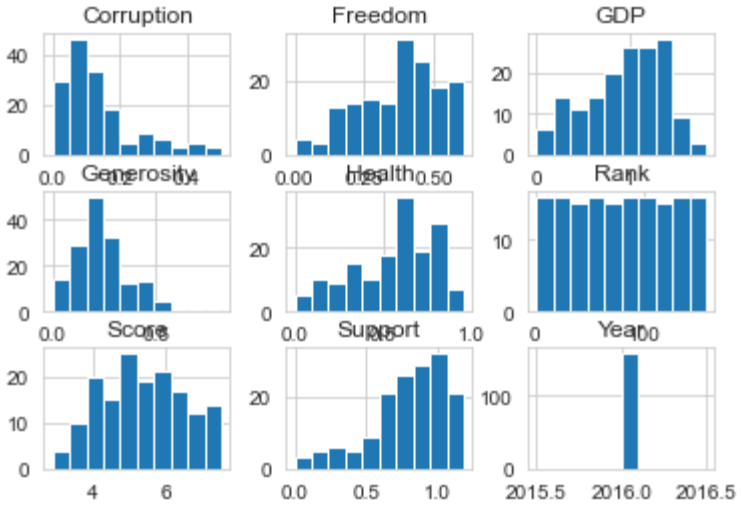
In [71]:
```
df2020.hist()
```

Out[71]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8C8A89D
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A1F761
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8C93C85
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A3A7CA
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CAE013
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8AFF84C
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8AFF85B
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CA90A6
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CA8A2B
0>]],
      dtype=object)
```



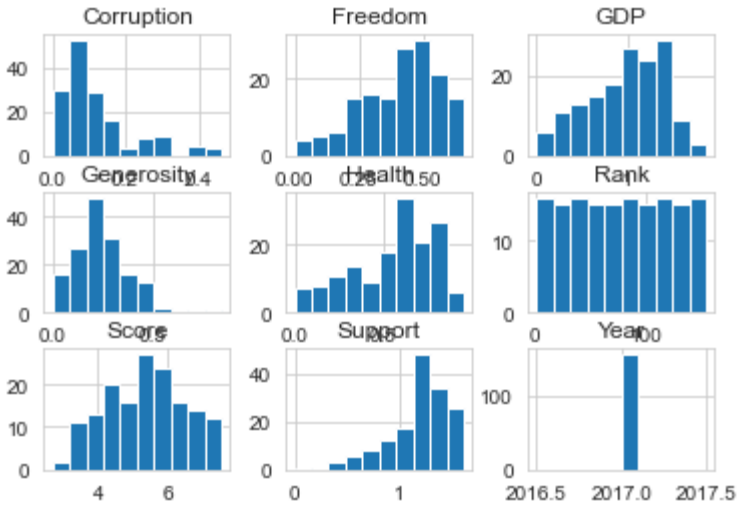
In [72]:
```
df2016.hist()
```

Out[72]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A19F820>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D058370>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CC877F0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CFC96D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8C98B910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D0C5670>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D042A00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8ACBD820>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B190850>]],
      dtype=object)
```



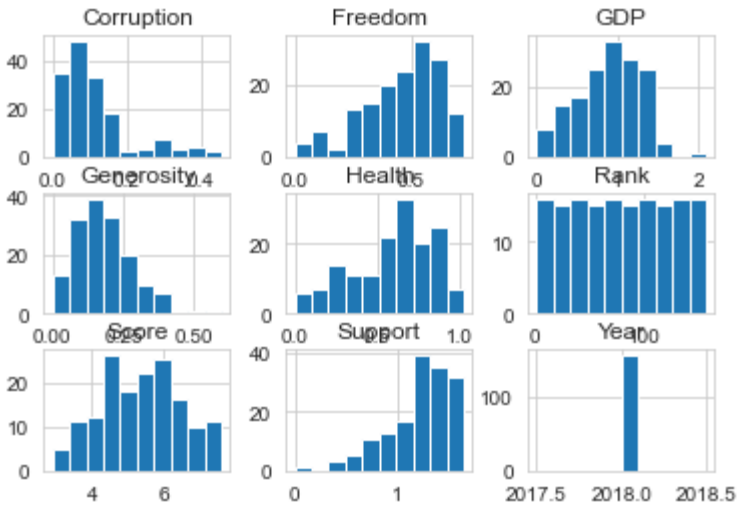
In [73]:
```
df2017.hist()
```

Out[73]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CFC06A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CFAABB0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A1460A0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8C8AD490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B0A18E0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B104C70>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B104D60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A22E0A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8AFBEF10>]],
      dtype=object)
```



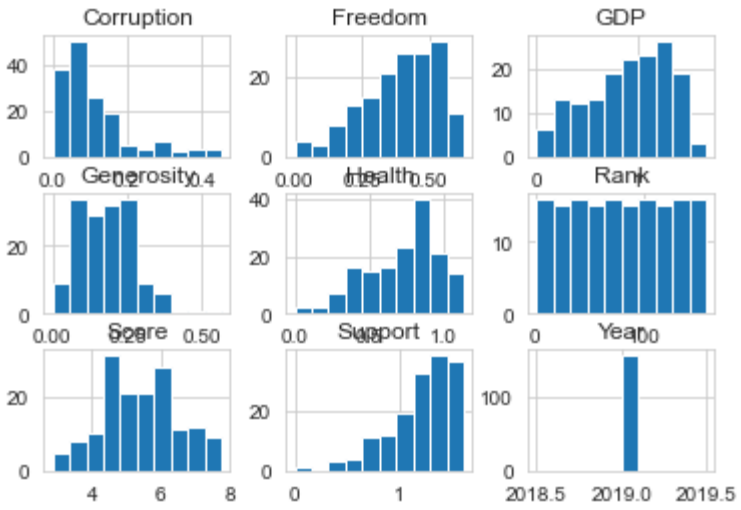
In [74]:
```
df2018.hist()
```

Out[74]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B0D0CD
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B0DA58
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A4CC8B
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B2110D
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CD787F
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B1CB04
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8B1CBFA
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CE917C
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CD1667
0>]],
      dtype=object)
```



In [75]:
```
df2019.hist()
```

Out[75]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CD168E
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D3E91C
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CFD694
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CFFA16
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CCCA88
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A6180A
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A62407
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A64B85
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A6AB70
0>]],
      dtype=object)
```



In [76]:
```
target = ['Top','Top-Mid', 'Low-Mid', 'Low' ]
target_n = [4, 3, 2, 1]
df2015["target"] = pd.qcut(df2015['Rank'], len(target), labels=target)
df2015["target_n"] = pd.qcut(df2015['Rank'], len(target), labels=target_n)
```

In [77]:
```
df = df2015.append([df2016,df2017,df2018,df2019])
```

In [78]:
```
df.isnull().any()
```

Out[78]:
```
Country       False
Region         True
Rank          False
Score         False
GDP           False
Support       False
Health        False
Freedom       False
Corruption     True
Generosity    False
Year          False
```

```
target          True
target_n        True
dtype: bool
```

In [79]:
```
df.Corruption.fillna((df.Corruption.mean()), inplace = True)
df.head()
```

Out[79]:

	Country	Region	Rank	Score	GDP	Support	Health	Freedom	Corruption	Generosity	Y
0	Switzerland	Western Europe	1	7.587	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2
1	Iceland	Western Europe	2	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2
2	Denmark	Western Europe	3	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2
3	Norway	Western Europe	4	7.522	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2
4	Canada	North America	5	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2

In [80]:
```
evaluation = pd.DataFrame({'Model':[],
                           'Details':[],
                           'Root Mean Squared Error (RMSE)': [],
                           'R-squared (training)': [],
                           'Adjusted R-squared (training)': [],
                           'R-squared (test)':[],
                           'Adjusted R-squared(test)':[],
                            '5-Fold Cross Validation':[]
                          })
```

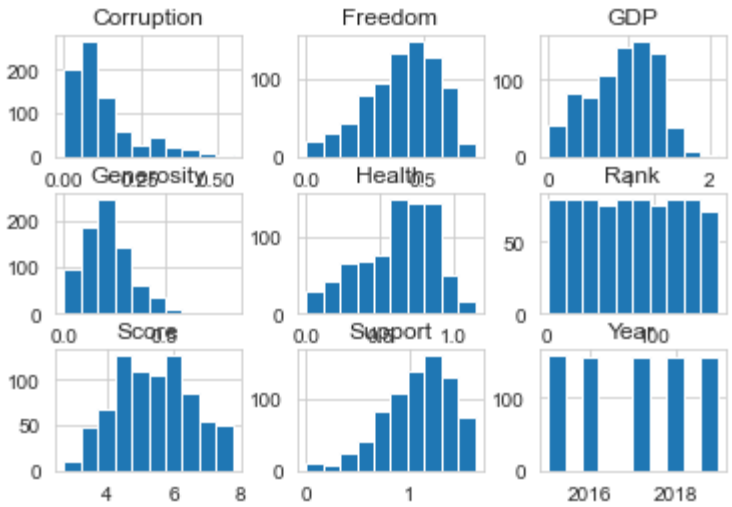
In [81]:
```
df.describe()
```

Out[81]:

	Rank	Score	GDP	Support	Health	Freedom	Corruption	Generosi
count	782.000000	782.000000	782.000000	782.000000	782.000000	782.000000	782.000000	782.0000
mean	78.698210	5.379018	0.916047	1.078392	0.612416	0.411091	0.125436	0.2185
std	45.182384	1.127456	0.407340	0.329548	0.248309	0.152880	0.105749	0.1223
min	1.000000	2.693000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	40.000000	4.509750	0.606500	0.869363	0.440183	0.309768	0.054250	0.1300
50%	79.000000	5.322000	0.982205	1.124735	0.647310	0.431000	0.091033	0.2019
75%	118.000000	6.189500	1.236187	1.327250	0.808000	0.531000	0.155861	0.2788
max	158.000000	7.769000	2.096000	1.644000	1.141000	0.724000	0.551910	0.8380

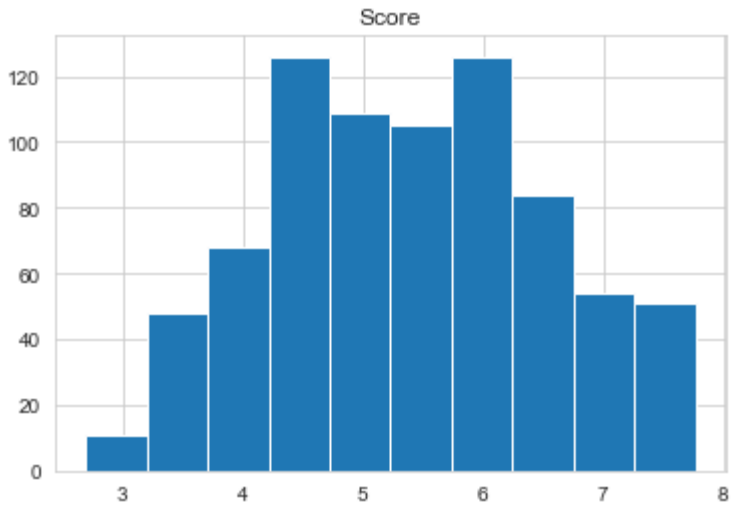
In [82]:
```
df.hist()
```

Out[82]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CB07F7
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A875C1
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A89F13
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8A8D652
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CCD397
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CCFED0
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CCFEDF
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8CE762E
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D10EAF
0>]],
      dtype=object)
```



In [83]:
```
df.hist('Score')
```

Out[83]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8D26513
0>]],
      dtype=object)
```



In [84]:
```
df
```

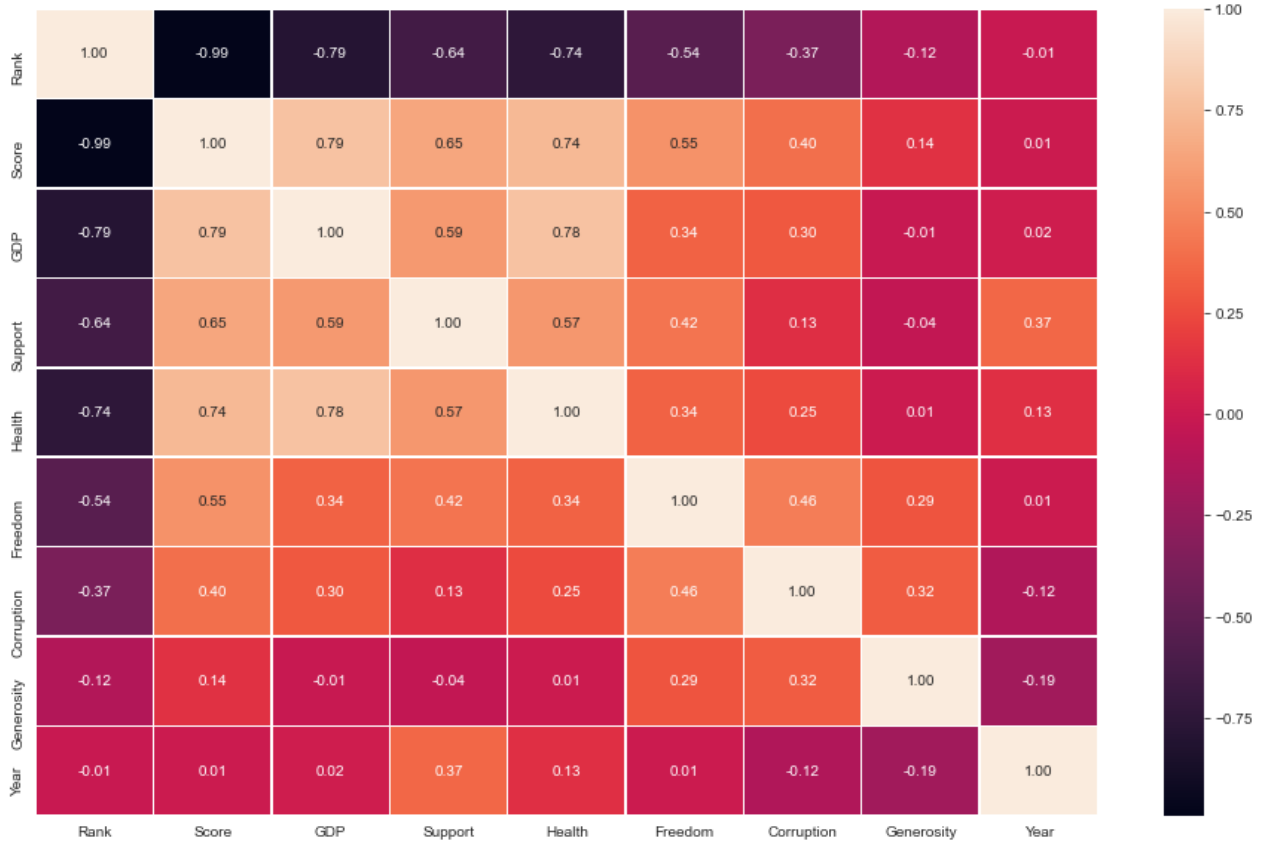
Out[84]:

	Country	Region	Rank	Score	GDP	Support	Health	Freedom	Corruption	Generosity
0	Switzerland	Western Europe	1	7.587	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	Iceland	Western Europe	2	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	Denmark	Western Europe	3	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	Norway	Western Europe	4	7.522	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	Canada	North America	5	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811
...	...	...	...	...	...	...	...	...	...	...
151	Rwanda	NaN	152	3.334	0.35900	0.71100	0.61400	0.55500	0.41100	0.21700
152	Tanzania	NaN	153	3.231	0.47600	0.88500	0.49900	0.41700	0.14700	0.27600
153	Afghanistan	NaN	154	3.203	0.35000	0.51700	0.36100	0.00000	0.02500	0.15800
154	Central African Republic	NaN	155	3.083	0.02600	0.00000	0.10500	0.22500	0.03500	0.23500
155	South Sudan	NaN	156	2.853	0.30600	0.57500	0.29500	0.01000	0.09100	0.20200

782 rows × 13 columns

In [85]:
```
f,ax=plt.subplots(figsize=(16,10))
sns.heatmap(df2015.corr(),annot=True, linewidth=.5,fmt='.2f',ax=ax)
plt.show()
```

In [86]:
```
f,ax=plt.subplots(figsize=(16,10))
sns.heatmap(df.corr(),annot=True, linewidth=.5,fmt='.2f',ax=ax)
plt.show()
```



In [87]:
```
px.scatter(df, x="GDP", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
lr = LinearRegression()
X_train = np.array(train_data['GDP'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['GDP'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))

rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['GDP']],df['Score'],cv=5).mean(),
'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

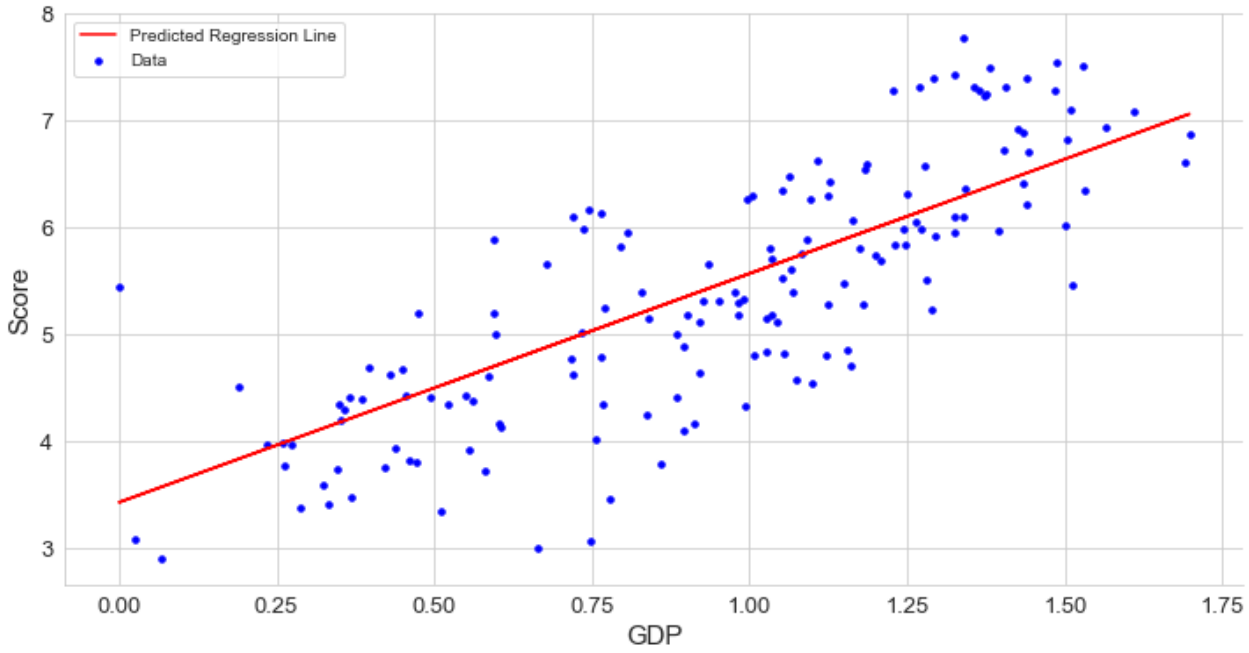
```
Average Score for Test Data: 5.388
Intercept: 3.430174973757425
Coefficient: [2.14115845]
```

Out[87]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.62	-	0.632	-	0.617

In [88]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("GDP", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()
```

```
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [89]:
```
px.scatter(df, x="Support", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
lr = LinearRegression()
X_train = np.array(train_data['Support'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['Support'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))

rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['Support']],df['Score'],cv=5).mea
n(),'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

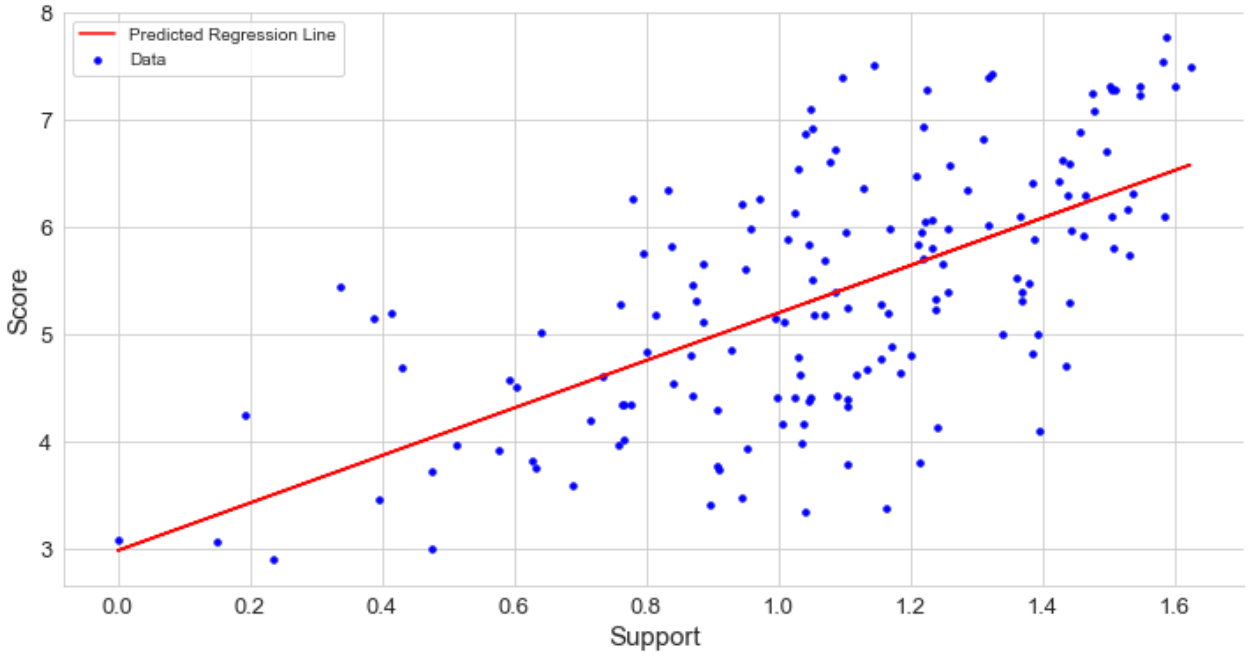
```
Average Score for Test Data: 5.388
```

```
Intercept: 2.987178261826246
Coefficient: [2.21482048]
```

Out[89]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.620	-	0.632	-	0.617
1	Simple Linear Regression	-	0.903	0.426	-	0.402	-	0.326

In [90]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("Support", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [91]:
```
px.scatter(df, x="Health", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
```

```
lr = LinearRegression()
X_train = np.array(train_data['Health'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['Health'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))

rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['Health']],df['Score'],cv=5).mean
(),'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

```
Average Score for Test Data: 5.388
Intercept: 3.3341420139695326
Coefficient: [3.35469024]
```

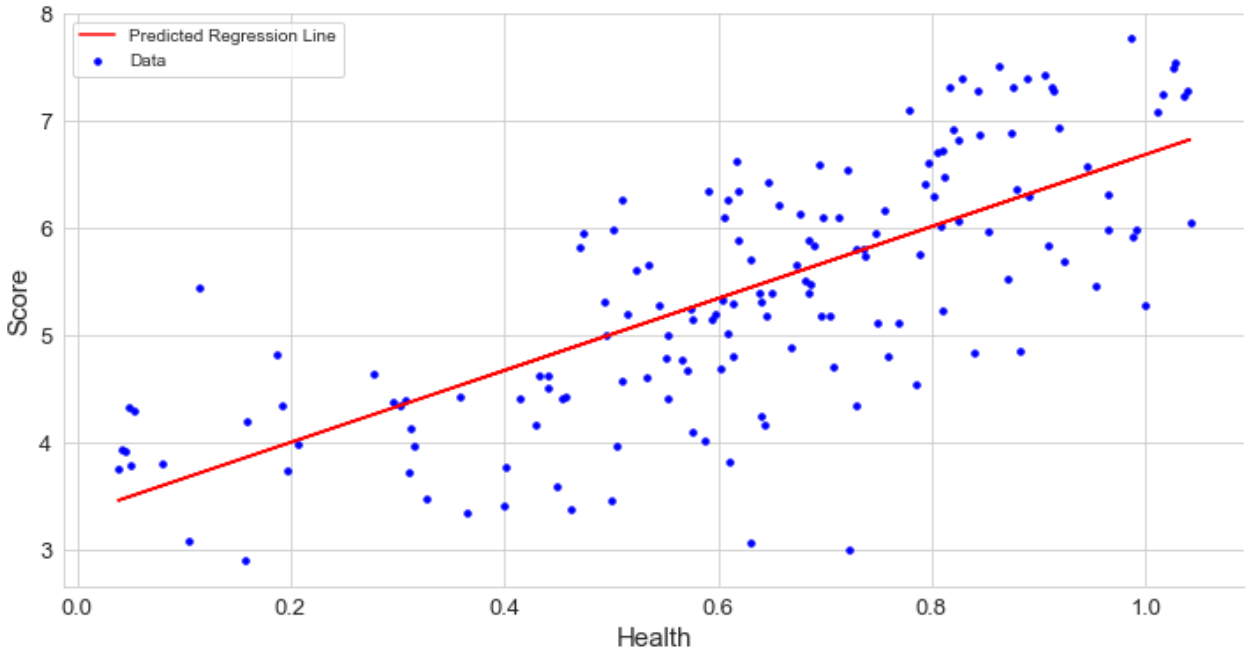
Out[91]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.620	-	0.632	-	0.617
1	Simple Linear Regression	-	0.903	0.426	-	0.402	-	0.326
2	Simple Linear Regression	-	0.795	0.555	-	0.536	-	0.530

In [92]:

```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("Health", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
```

```
plt.legend()
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [93]:

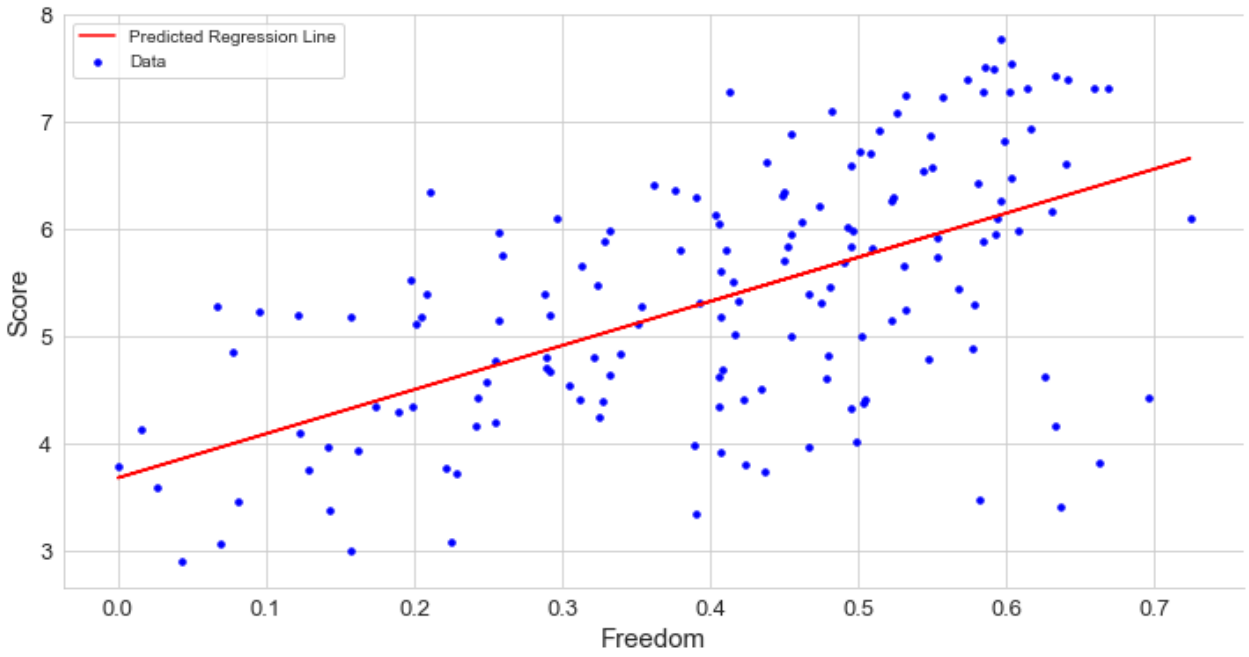
```
px.scatter(df, x="Freedom", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
lr = LinearRegression()
X_train = np.array(train_data['Freedom'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['Freedom'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))

rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['Freedom']],df['Score'],cv=5).mea
n(),'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

```
Average Score for Test Data: 5.388
Intercept: 3.683005610778345
Coefficient: [4.11588367]
```

Out[93]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.620	-	0.632	-	0.617
1	Simple Linear Regression	-	0.903	0.426	-	0.402	-	0.326
2	Simple Linear Regression	-	0.795	0.555	-	0.536	-	0.530
3	Simple Linear Regression	-	0.967	0.301	-	0.313	-	0.295

In [94]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("Freedom", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [95]:
```
px.scatter(df, x="Generosity", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
lr = LinearRegression()
X_train = np.array(train_data['Generosity'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['Generosity'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))

rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['Generosity']],df['Score'],cv=5).
mean(),'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

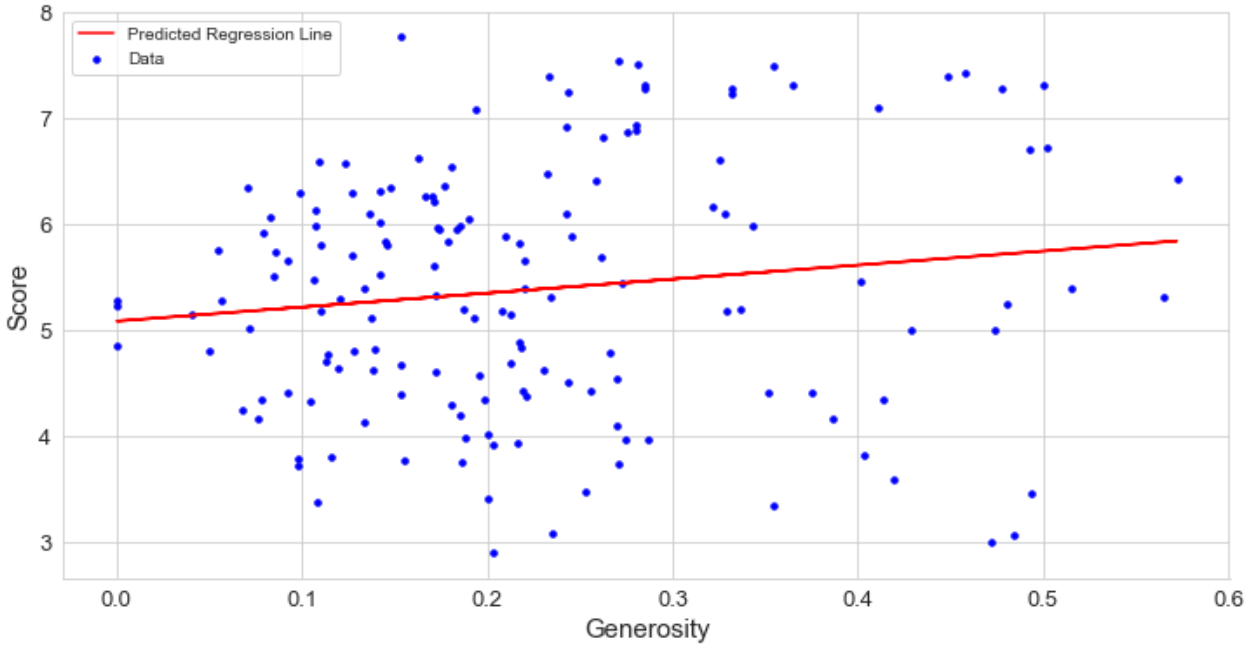
```
Average Score for Test Data: 5.388
Intercept: 5.089485375227209
Coefficient: [1.32184138]
```

Out[95]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.620	-	0.632	-	0.617
1	Simple Linear Regression	-	0.903	0.426	-	0.402	-	0.326
2	Simple Linear Regression	-	0.795	0.555	-	0.536	-	0.530
3	Simple Linear Regression	-	0.967	0.301	-	0.313	-	0.295

4	Simple Linear Regression	-	1.160	0.021	-	0.012	-	0.017

In [96]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("Generosity", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [97]:
```
px.scatter(df, x="Corruption", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
train_data, test_data = train_test_split(df, train_size = 0.8, random_stat
e = 3)
lr = LinearRegression()
X_train = np.array(train_data['Corruption'],
                   dtype = pd.Series).reshape(-1,1)
y_train = np.array(train_data['Score'], dtype = pd.Series)
lr.fit(X_train, y_train)
X_test = np.array(test_data['Corruption'],
                    dtype = pd.Series).reshape(-1,1)
y_test = np.array(test_data['Score'], dtype = pd.Series)
pred = lr.predict(X_test)

rmsesm = float(format(np.sqrt(mean_squared_error(y_test,pred)),'.3f'))
```

```
rtrsm = float(format(lr.score(X_train, y_train),'.3f'))

rtesm = float(format(lr.score(X_test, y_test),'.3f'))
cv = float(format(cross_val_score(lr,df[['Corruption']],df['Score'],cv=5).
mean(),'.3f'))
print ("Average Score for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))
r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Linear Regression','-',rmsesm,rtrsm,'-',rtesm
,'-',cv]
evaluation
```

```
Average Score for Test Data: 5.388
Intercept: 4.872968925819756
Coefficient: [4.1304732]
```

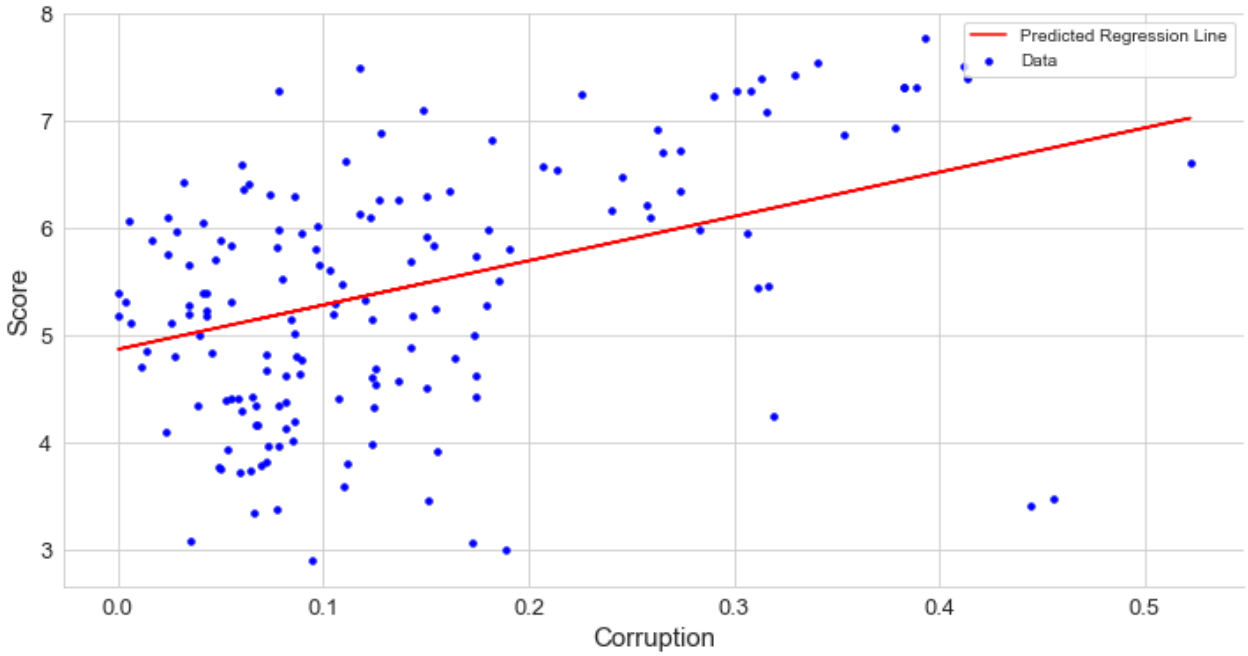
Out[97]:

	Model	Details	Root Mean Squared Error (RMSE)	R-squared (training)	Adjusted R-squared (training)	R-squared (test)	Adjusted R-squared(test)	5-Fold Cross Validation
0	Simple Linear Regression	-	0.708	0.620	-	0.632	-	0.617
1	Simple Linear Regression	-	0.903	0.426	-	0.402	-	0.326
2	Simple Linear Regression	-	0.795	0.555	-	0.536	-	0.530
3	Simple Linear Regression	-	0.967	0.301	-	0.313	-	0.295
4	Simple Linear Regression	-	1.160	0.021	-	0.012	-	0.017
5	Simple Linear Regression	-	1.046	0.148	-	0.196	-	0.155

In [98]:

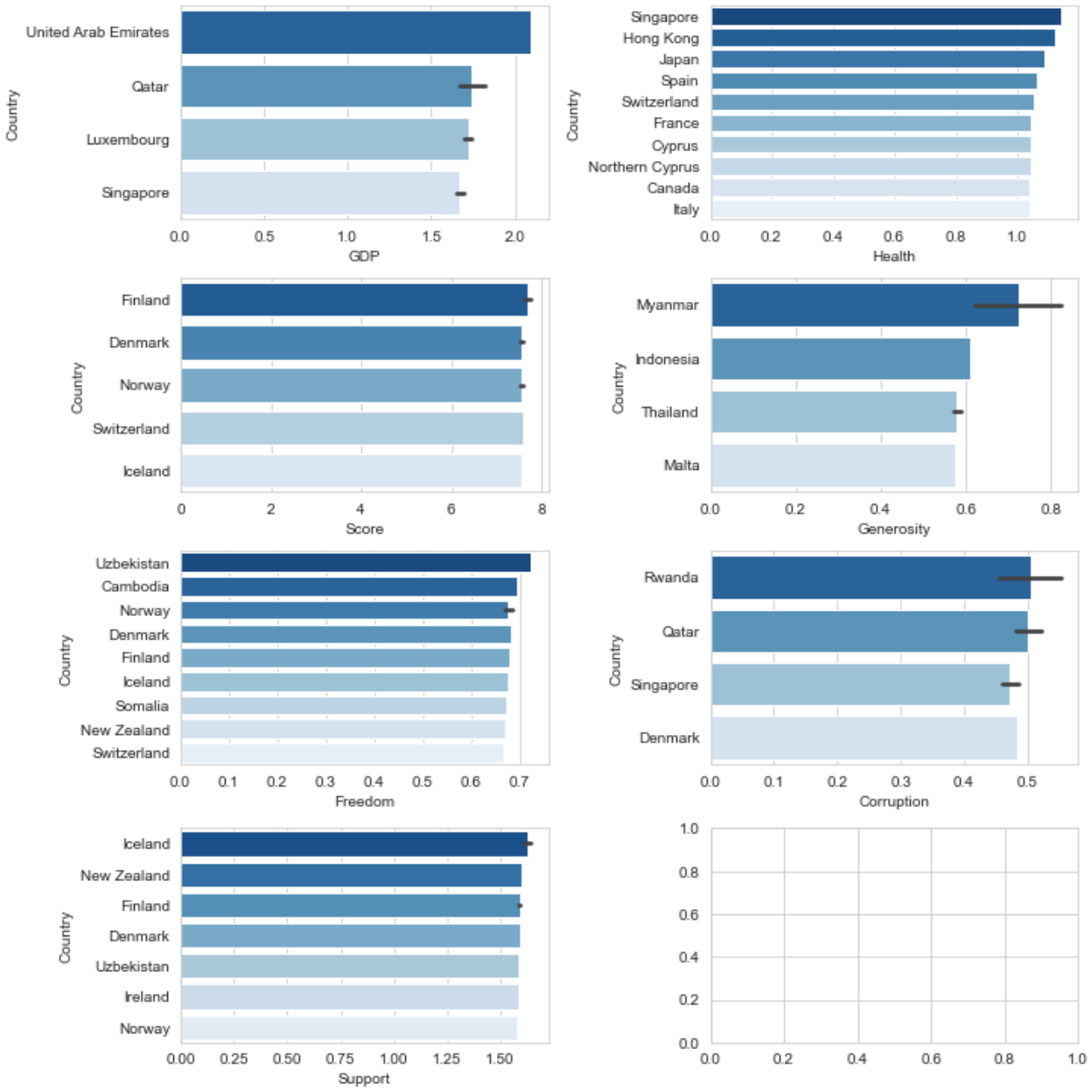
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,6))
plt.scatter(X_test,y_test,color='blue',label="Data", s = 12)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression
 Line")
plt.xlabel("Corruption", fontsize=15)
plt.ylabel("Score", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()
```

```
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```



In [103]:
```
fig, axes = plt.subplots(nrows=4, ncols=2,constrained_layout=True,figsize=
(10,10))
sns.barplot(x='GDP',y='Country',
                        data=df.nlargest(10,'GDP'),
                        ax=axes[0,0],palette="Blues_r")
sns.barplot(x='Health' ,y='Country',
                        data=df.nlargest(10,'Health'),
                        ax=axes[0,1],palette='Blues_r')
sns.barplot(x='Score' ,y='Country',
                        data=df.nlargest(10,'Score'),
                        ax=axes[1,0],palette='Blues_r')
sns.barplot(x='Generosity' ,y='Country',
                        data=df.nlargest(10,'Generosity'),
                        ax=axes[1,1],palette='Blues_r')
sns.barplot(x='Freedom' ,y='Country',
                        data=df.nlargest(10,'Freedom'),
                        ax=axes[2,0],palette='Blues_r')
sns.barplot(x='Corruption' ,y='Country',
                        data=df.nlargest(10,'Corruption'),
                        ax=axes[2,1],palette='Blues_r')
sns.barplot(x='Support' ,y='Country',
                        data=df.nlargest(10,'Support'),
                        ax=axes[3,0],palette='Blues_r')
```

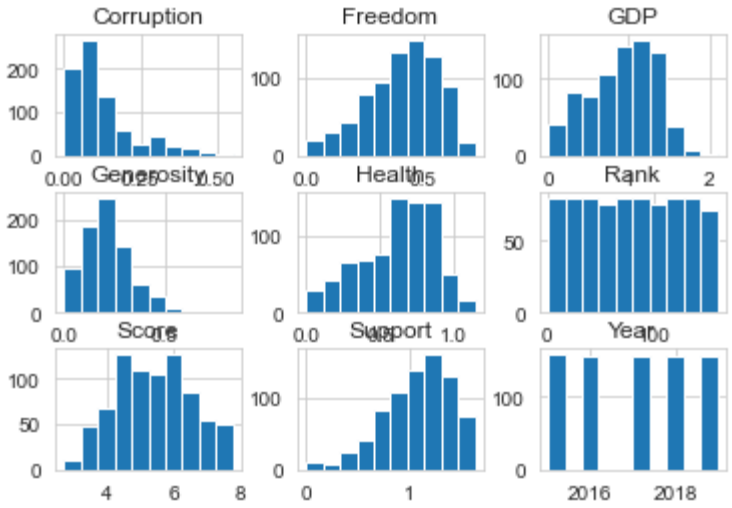
Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x23d8f1daa90>



In [100]:
```
df.hist()
```

Out[100]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8FE5D9A
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8E71A28
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8E7026D
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8F9C5AF
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8E591F7
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8F5EE34
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8F5EE43
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8F5E28E
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023D8E68E13
```

```
0>]],
      dtype=object)
```



In [101]:
```
data = dict(type = 'choropleth',
            locations = df['Country'],
            locationmode = 'country names',
            z = df['Score'],
            text = df['Country'],
            colorscale = 'YlGnBu',
            autocolorscale=False,
            colorbar={"title":"Happy"})
layout = dict(title = 'Happiest Place on Earth',
              geo = dict(showframe = False,
                         projection = {'type': 'equirectangular'}))
choromap3= pgo.Figure(data = [data], layout=layout)
iplot(choromap3)
```

In [ ]: