In [1]:
```
import pandas as pd           # importing modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data2015 = pd.read_csv(r'F:\DataScience\Data\happiness\\2015.csv')    # importing data
data2016 = pd.read_csv(r'F:\DataScience\Data\happiness\\2016.csv')
data2017 = pd.read_csv(r'F:\DataScience\Data\happiness\\2017.csv')
data2018 = pd.read_csv(r'F:\DataScience\Data\happiness\\2018.csv')
data2019 = pd.read_csv(r'F:\DataScience\Data\happiness\\2019.csv')
data2020 = pd.read_csv(r'F:\DataScience\Data\happiness\\2020.csv',sep = ';')
```

In [2]:
```
# 2015 - unification of column names, deletion of unnecessary columns and setting the index

dictionary = {'Happiness Rank':'Overall rank','Happiness Score':'Score','Economy (GDP per Capita)':'GDP per capita', 'Health (Life Expectancy)':'Healthy life expectancy','Freedom':'
data2015.rename(dictionary, axis = 1, inplace = True)
del data2015['Region']
del data2015['Standard Error']
del data2015['Dystopia Residual']
data2015.set_index('Country', inplace = True)
data2015.head()
```

Out[2]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
Switzerland	1	7.587	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
Iceland	2	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
Denmark	3	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
Norway	4	7.522	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
Canada	5	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811

In [3]:
```
# 2016 -  unification of column names, deletion of unnecessary columns and setting the index

dictionary = {'Happiness Rank':'Overall rank','Happiness Score':'Score','Economy (GDP per Capita)':'GDP per capita', 'Health (Life Expectancy)':'Healthy life expectancy','Freedom':'
data2016.rename(dictionary, axis = 1, inplace = True)
del data2016['Region']
del data2016['Lower Confidence Interval']
del data2016['Upper Confidence Interval']
del data2016['Dystopia Residual']
data2016.set_index('Country', inplace = True)
data2016.head()
```

Out[3]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
Denmark	1	7.526	1.44178	1.16374	0.79504	0.57941	0.44453	0.36171
Switzerland	2	7.509	1.52733	1.14524	0.86303	0.58557	0.41203	0.28083
Iceland	3	7.501	1.42666	1.18326	0.86733	0.56624	0.14975	0.47678
Norway	4	7.498	1.57744	1.12690	0.79579	0.59609	0.35776	0.37895
Finland	5	7.413	1.40598	1.13464	0.81091	0.57104	0.41004	0.25492

In [4]:
```
# 2017 -  unification of column names, deletion of unnecessary columns and setting the index

dictionary = {'Happiness.Rank':'Overall rank','Happiness.Score':'Score','Economy..GDP.per.Capita.':'GDP per capita', 'Health..Life.Expectancy.':'Healthy life expectancy','Freedom':'
data2017.rename(dictionary, axis = 1, inplace = True)
del data2017['Whisker.high']
del data2017['Whisker.low']
del data2017['Dystopia.Residual']
data2017.set_index('Country', inplace = True)
data2017.head()
```

Out[4]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
Norway	1	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964
Denmark	2	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770
Iceland	3	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527
Switzerland	4	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007
Finland	5	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612

In [5]:
```
# 2018 -  unification of column names and setting the index

dictionary = {'Country or region':'Country'}
data2018.rename(dictionary, axis = 1, inplace = True)
data2018.set_index('Country', inplace = True)
data2018.head()
```

Out[5]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
Finland	1	7.632	1.305	1.592	0.874	0.681	0.202	0.393
Norway	2	7.594	1.456	1.582	0.861	0.686	0.286	0.340
Denmark	3	7.555	1.351	1.590	0.868	0.683	0.284	0.408
Iceland	4	7.495	1.343	1.644	0.914	0.677	0.353	0.138
Switzerland	5	7.487	1.420	1.549	0.927	0.660	0.256	0.357

In [6]:
```
# 2019 -  unification of column names and setting the index

dictionary = {'Country or region':'Country'}
data2019.rename(dictionary, axis = 1, inplace = True)
data2019.set_index('Country', inplace = True)
data2019.head()
```

Out[6]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
Finland	1	7.769	1.340	1.587	0.986	0.596	0.153	0.393
Denmark	2	7.600	1.383	1.573	0.996	0.592	0.252	0.410
Norway	3	7.554	1.488	1.582	1.028	0.603	0.271	0.341
Iceland	4	7.494	1.380	1.624	1.026	0.591	0.354	0.118
Netherlands	5	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [7]:
```
# 2020 -  unification of column names and setting the index

dictionary = {'Country or region':'Country'}
data2020.rename(dictionary, axis = 1, inplace = True)
data2020.set_index('Country', inplace = True)
data2020.head()
```

Out[7]:

Country	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
Finland	1	7.809	1.285	1.500	0.961	0.662	0.160	0.478
Denmark	2	7.646	1.327	1.503	0.979	0.665	0.243	0.495
Switzerland	3	7.560	1.391	1.472	1.041	0.629	0.269	0.408
Iceland	4	7.504	1.327	1.548	1.001	0.662	0.362	0.145
Norway	5	7.488	1.424	1.495	1.008	0.670	0.288	0.434

In [8]:
```
# adding column with info about year

data2015.insert(loc = 8, column = 'Year', value = '2015')
data2016.insert(loc = 8, column = 'Year', value = '2016')
data2017.insert(loc = 8, column = 'Year', value = '2017')
data2018.insert(loc = 8, column = 'Year', value = '2018')
data2019.insert(loc = 8, column = 'Year', value = '2019')
data2020.insert(loc = 8, column = 'Year', value = '2020')
```

In [9]:
```
# connecting DataFrames from all years to one Frame

data = data2015.append(data2016).append(data2017).append(data2018).append(data2019).append(data2020)
len(data)
```

Out[9]: 935

In [10]:
```
# setting multiindex as "Year" and "Country", sorting both indexes ascending, replacing 'NaN' ---> 0

data.reset_index(inplace = True)
data.set_index(['Country','Year'], inplace = True)
data.sort_index(inplace = True)
data.fillna(value = 0, inplace = True)
```

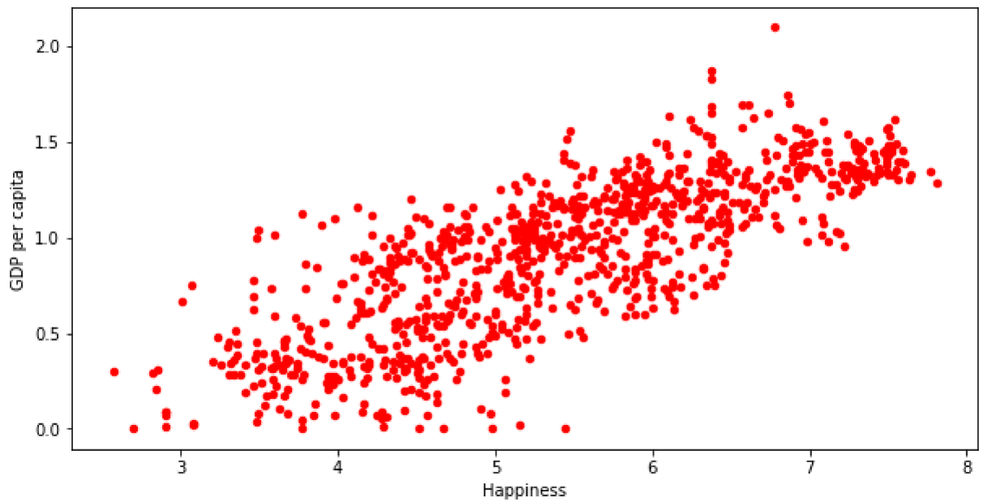
In [11]:
```
# rounding numbers to 3 decimal places and checking a data

col_names = ['Overall rank', 'Score', 'GDP per capita', 'Social support','Healthy life expectancy',
             'Freedom to make life choices','Perceptions of corruption', 'Generosity']
data[col_names] = round(data[col_names],3)
data.head(12)
```

Out[11]:

Country	Year	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
Afghanistan	**2015**	153	3.575	0.320	0.303	0.303	0.234	0.097	0.365
	2016	154	3.360	0.382	0.110	0.173	0.164	0.071	0.313
	2017	141	3.794	0.401	0.582	0.181	0.106	0.061	0.312
	2018	145	3.632	0.332	0.537	0.255	0.085	0.036	0.191
	2019	154	3.203	0.350	0.517	0.361	0.000	0.025	0.158
	2020	153	2.567	0.301	0.356	0.266	0.000	0.001	0.135
Albania	**2015**	95	4.959	0.879	0.804	0.813	0.357	0.064	0.143
	2016	109	4.655	0.955	0.502	0.730	0.319	0.053	0.168
	2017	109	4.644	0.996	0.804	0.731	0.381	0.040	0.201
	2018	112	4.586	0.916	0.817	0.790	0.419	0.032	0.149
	2019	107	4.719	0.947	0.848	0.874	0.383	0.027	0.178
	2020	105	4.883	0.907	0.830	0.846	0.462	0.025	0.171

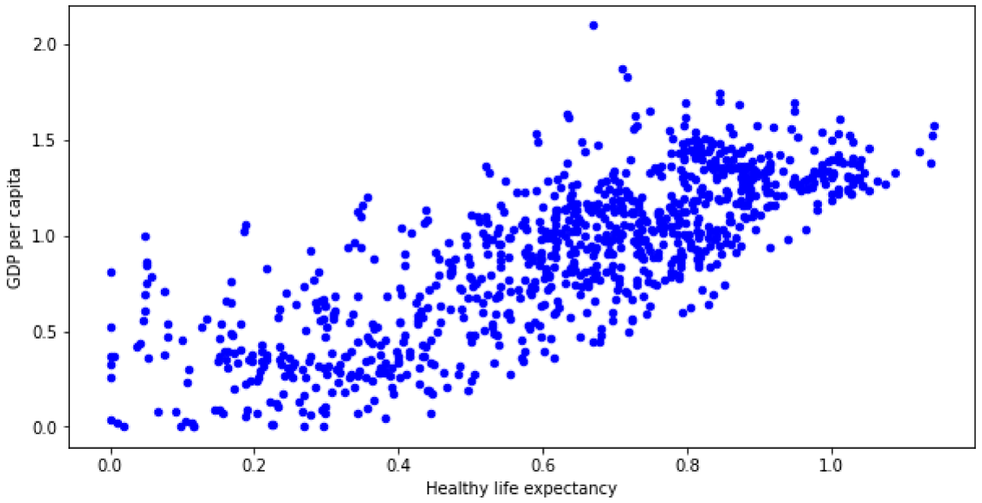
In [12]:
```
# Does money give happiness?

data.plot(kind = 'scatter', x = 'Score', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Happiness', figsize = (10,5), color = 'r')

# World rather says...YES.
```

Out[12]: <AxesSubplot:xlabel='Happiness', ylabel='GDP per capita'>



In [13]:
```
# Can we buy health?

data.plot(kind = 'scatter', x = 'Healthy life expectancy', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Healthy life expectancy', figsize = (10,5), color = 'b')

# World also suggest answer "YES".
```

Out[13]: <AxesSubplot:xlabel='Healthy life expectancy', ylabel='GDP per capita'>



In [14]:
```
# OK, so lets find a place on Earth, where we can be happy without big money (GDP) influence :)

data['Happy/GDP Ratio'] = round(data['Score'] / data['GDP per capita'], 3)    # calculating "Happiness to GDP ratio"

hasGDP = data['GDP per capita'] > 0.2    # can be low GDP, but lets say must be at least 0.2

ScoreOver7 = data['Score'] > 7.0           # we are looking for really happy countries - level at least 7/10

data[hasGDP & ScoreOver7].sort_values(by = 'Happy/GDP Ratio', ascending = False).head() # Finding of results

# All top5 countries from Latin America!
```

Out[14]:

Country	Year	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity	Happy/GDP Ratio
Costa Rica	**2015**	12	7.226	0.956	1.238	0.860	0.634	0.106	0.255	7.559
	2020	15	7.121	0.981	1.375	0.940	0.645	0.096	0.131	7.259
Mexico	**2015**	14	7.187	1.021	0.915	0.814	0.482	0.213	0.141	7.039
Costa Rica	**2018**	13	7.072	1.010	1.459	0.817	0.632	0.101	0.143	7.002
	2019	12	7.167	1.034	1.441	0.963	0.558	0.093	0.144	6.931

In [15]:
```
# Now, on the contrary, let's find some countries, where happinness is mainly based on money.

hasBigGDP = data['GDP per capita'] > data['GDP per capita'].mean()    # GDP above average

ScoreOver4 = data['Score'] > 4           # Happiness can be low, but lets say must be at least 4/10

data[hasBigGDP & ScoreOver4].sort_values(by = 'Happy/GDP Ratio', ascending = True).head()

# Top3 results comes from Arabian countries, 4/5 from Asia.
```

Out[15]:

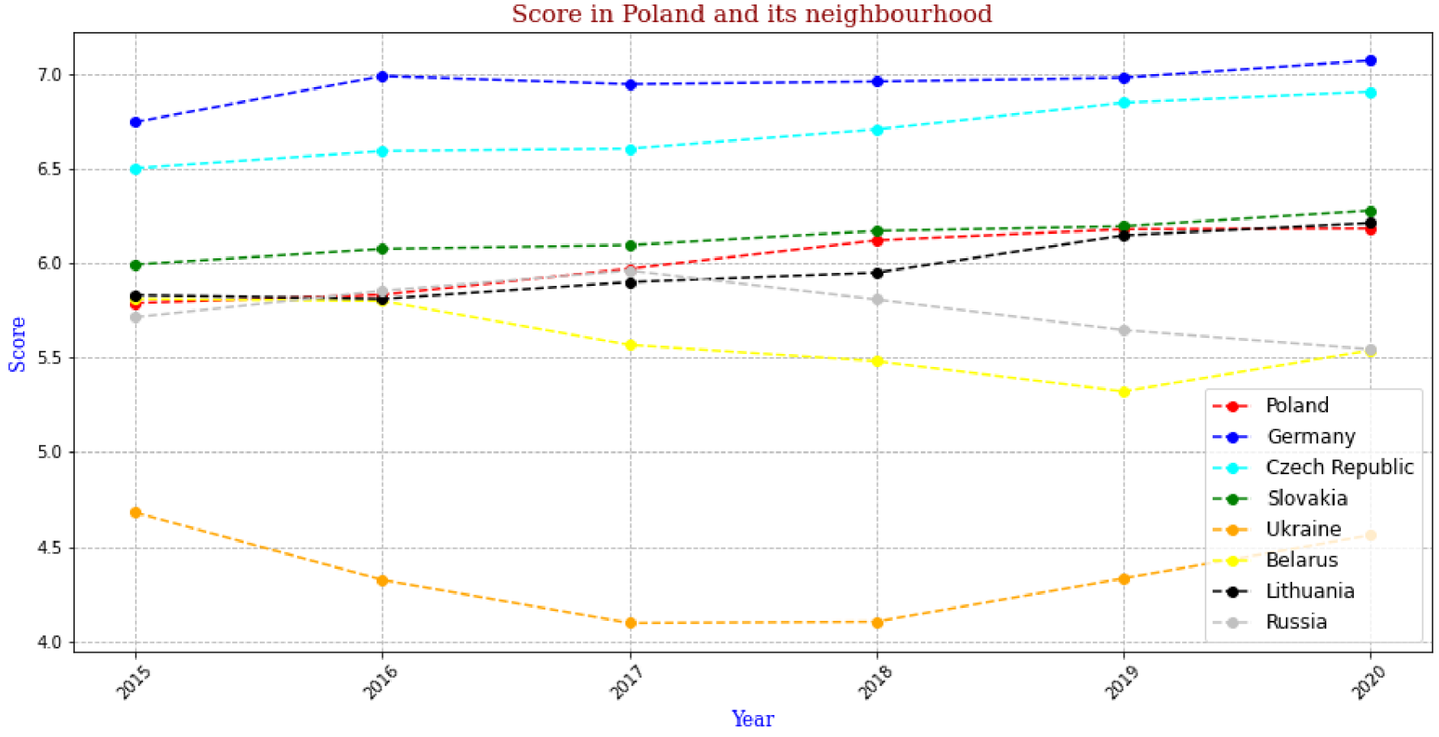
Country	Year	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity	Happy/GDP Ratio
United Arab Emirates	**2018**	20	6.774	2.096	0.776	0.670	0.284	0.000	0.186	3.232
Qatar	**2017**	35	6.375	1.871	1.274	0.710	0.604	0.439	0.330	3.407
	2016	36	6.375	1.824	0.880	0.717	0.567	0.480	0.324	3.495
Hong Kong S.A.R., China	**2017**	71	5.472	1.552	1.263	0.943	0.491	0.294	0.374	3.526
Gabon	**2016**	134	4.121	1.159	0.724	0.349	0.281	0.093	0.062	3.556

In [16]:
```
#  removing column 'Happy/GDP Ratio' and reindex
del data['Happy/GDP Ratio']
data.reset_index(inplace = True)
data.set_index('Country', inplace = True)
data.head()
```

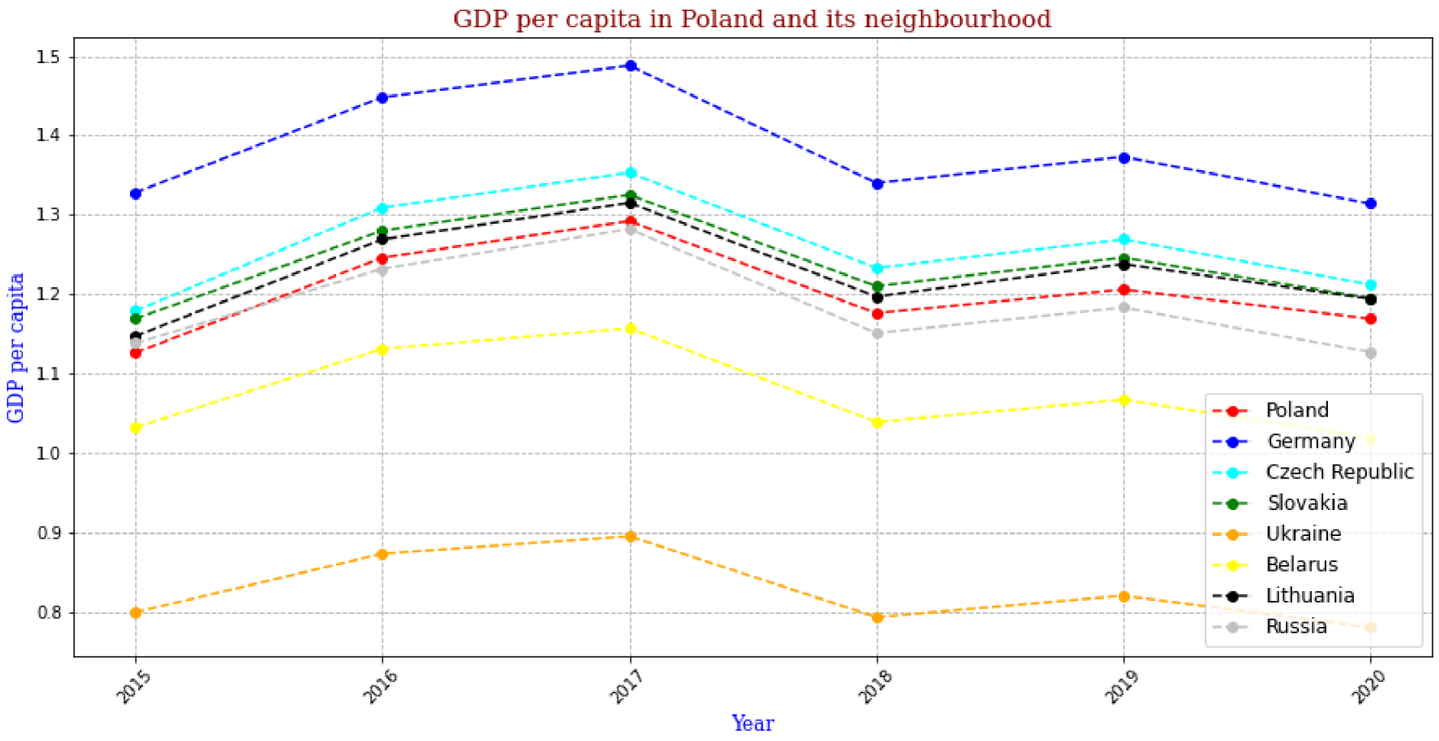
Out[16]:

Country	Year	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
Afghanistan	2015	153	3.575	0.320	0.303	0.303	0.234	0.097	0.365
Afghanistan	2016	154	3.360	0.382	0.110	0.173	0.164	0.071	0.313
Afghanistan	2017	141	3.794	0.401	0.582	0.181	0.106	0.061	0.312
Afghanistan	2018	145	3.632	0.332	0.537	0.255	0.085	0.036	0.191
Afghanistan	2019	154	3.203	0.350	0.517	0.361	0.000	0.025	0.158

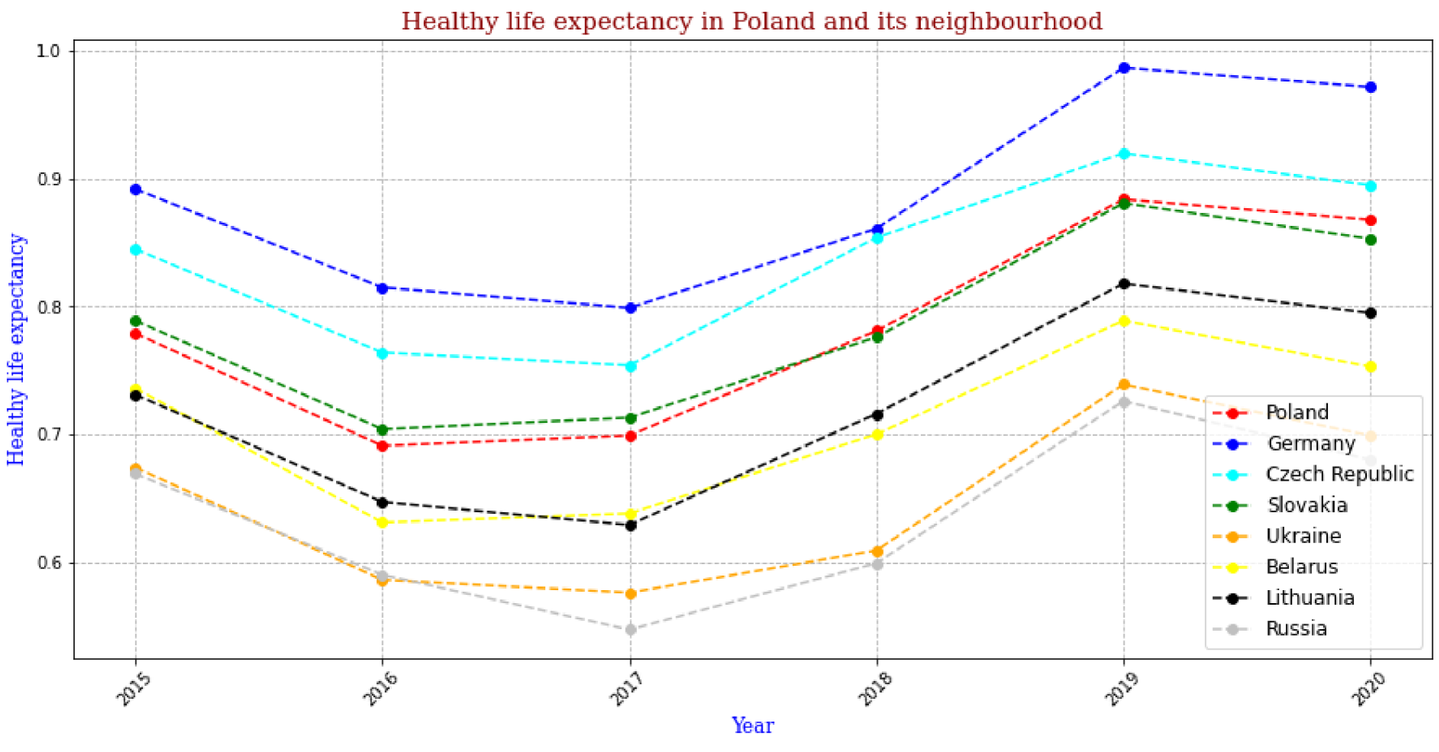
In [17]:
```
# defining a function to represent the selected column for Poland its neighbours

def showCol(colname):
    plt.figure( figsize = (15,7))
    plt.plot(data.loc['Poland','Year'], data.loc['Poland',colname], color = 'red', label = 'Poland', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Germany','Year'],data.loc['Germany',colname], color = 'blue', label = 'Germany', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Czech Republic','Year'],data.loc['Czech Republic',colname], color = 'cyan', label = 'Czech Republic', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Slovakia','Year'],data.loc['Slovakia',colname], color = 'green', label = 'Slovakia', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Ukraine','Year'],data.loc['Ukraine',colname], color = 'orange', label = 'Ukraine', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Belarus','Year'],data.loc['Belarus',colname], color = 'yellow', label = 'Belarus', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Lithuania','Year'],data.loc['Lithuania',colname], color = 'black', label = 'Lithuania', marker = 'o', linestyle = 'dashed')
    plt.plot(data.loc['Russia','Year'],data.loc['Russia',colname], color = 'silver', label = 'Russia', marker = 'o', linestyle = 'dashed')

    font1 = {'family':'serif','color':'blue','size':12}
    font2 = {'family':'serif','color':'darkred','size':15}

    plt.xlabel('Year', fontdict = font1 )
    plt.ylabel(colname, fontdict = font1)
    plt.title(colname + ' in Poland and its neighbourhood', fontdict = font2)
    plt.legend(loc = 'lower right', fontsize = 12)
    plt.grid(linestyle = '--')
    plt.xticks(rotation = 45)
    plt.show()

# calling the function for "Score"

showCol('Score')
```



In [18]:
```
# calling the function for "GDP per capita"

showCol('GDP per capita')
```

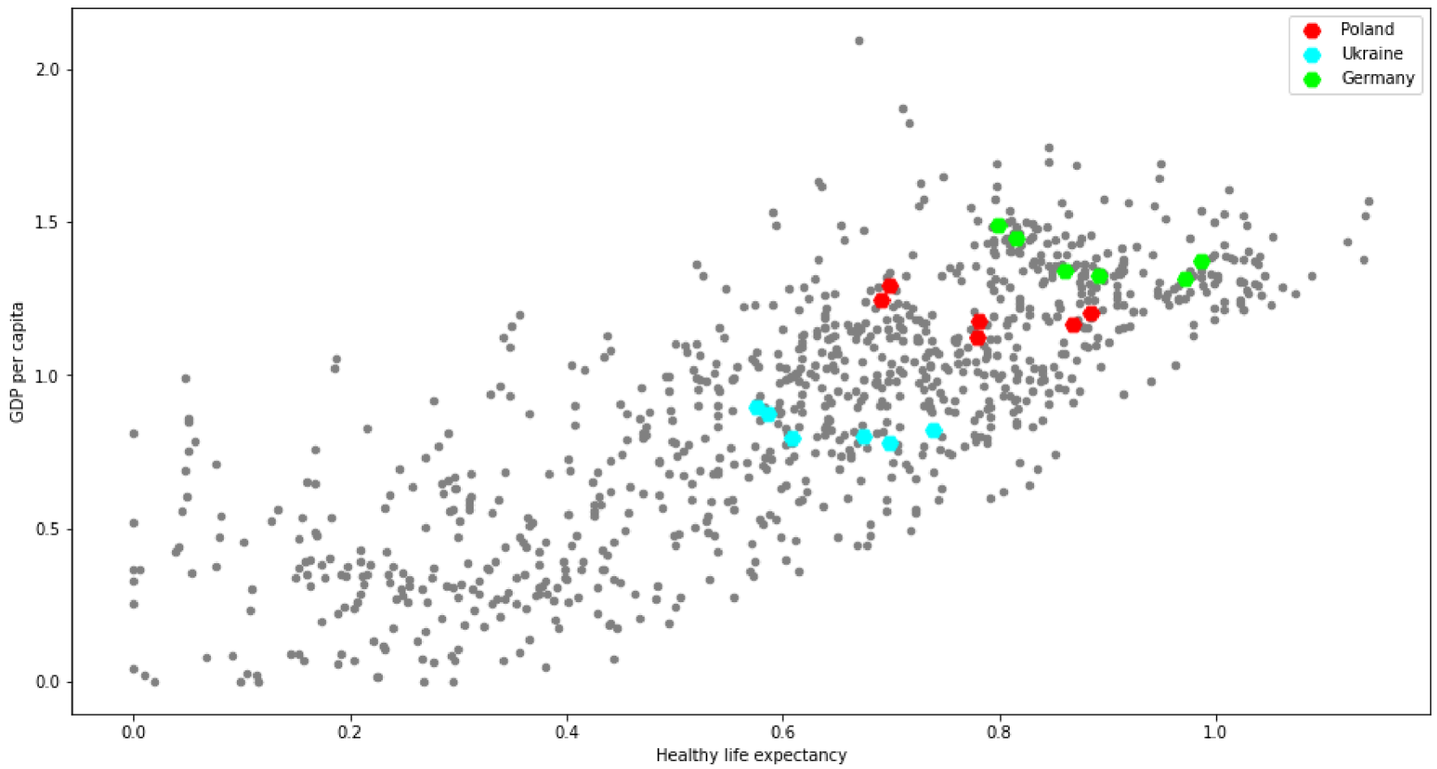


In [19]:
```
# calling the function for "Healthy life expectancy"

showCol('Healthy life expectancy')
```



In [20]:
```
# calling the function for "Freedom to make life choices"

showCol('Freedom to make life choices')
```

In [21]:
```
# "Healthy life expectancy" and "GDP per capita" for Poland, Germany and Ukraine compared to the rest of the world.

ALL = data.plot(kind = 'scatter', x = 'Healthy life expectancy', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Healthy life expectancy', color = 'gray')

PL = data.loc['Poland'].plot(kind = 'scatter', x = 'Healthy life expectancy', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Healthy life expectancy', color = '#ff0000', ax = ALL, label = 'Poland', marker = 'H', s = 100)

UKR = data.loc['Ukraine'].plot(kind = 'scatter', x = 'Healthy life expectancy', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Healthy life expectancy', color = '#00ffff', ax = PL, label = 'Ukraine', marker = 'H', s = 100)

DEU = data.loc['Germany'].plot(kind = 'scatter', x = 'Healthy life expectancy', y = 'GDP per capita',
          ylabel = 'GDP per capita', xlabel = 'Healthy life expectancy', figsize = (15,8), color = '#00ff00', ax = UKR, label = 'Germany', marker = 'H', s = 100)
```



In [22]:
```
data
```

Out[22]:

Country	Year	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
Afghanistan	2015	153	3.575	0.320	0.303	0.303	0.234	0.097	0.365
Afghanistan	2016	154	3.360	0.382	0.110	0.173	0.164	0.071	0.313
Afghanistan	2017	141	3.794	0.401	0.582	0.181	0.106	0.061	0.312
Afghanistan	2018	145	3.632	0.332	0.537	0.255	0.085	0.036	0.191
Afghanistan	2019	154	3.203	0.350	0.517	0.361	0.000	0.025	0.158
...	...	...	...	...	...	...	...	...	...
Zimbabwe	2016	131	4.193	0.350	0.715	0.160	0.254	0.086	0.185
Zimbabwe	2017	138	3.875	0.376	1.083	0.197	0.336	0.095	0.189
Zimbabwe	2018	144	3.692	0.357	1.094	0.248	0.406	0.099	0.132
Zimbabwe	2019	146	3.663	0.366	1.114	0.433	0.361	0.089	0.151
Zimbabwe	2020	151	3.299	0.426	1.048	0.375	0.377	0.081	0.151

935 rows × 9 columns

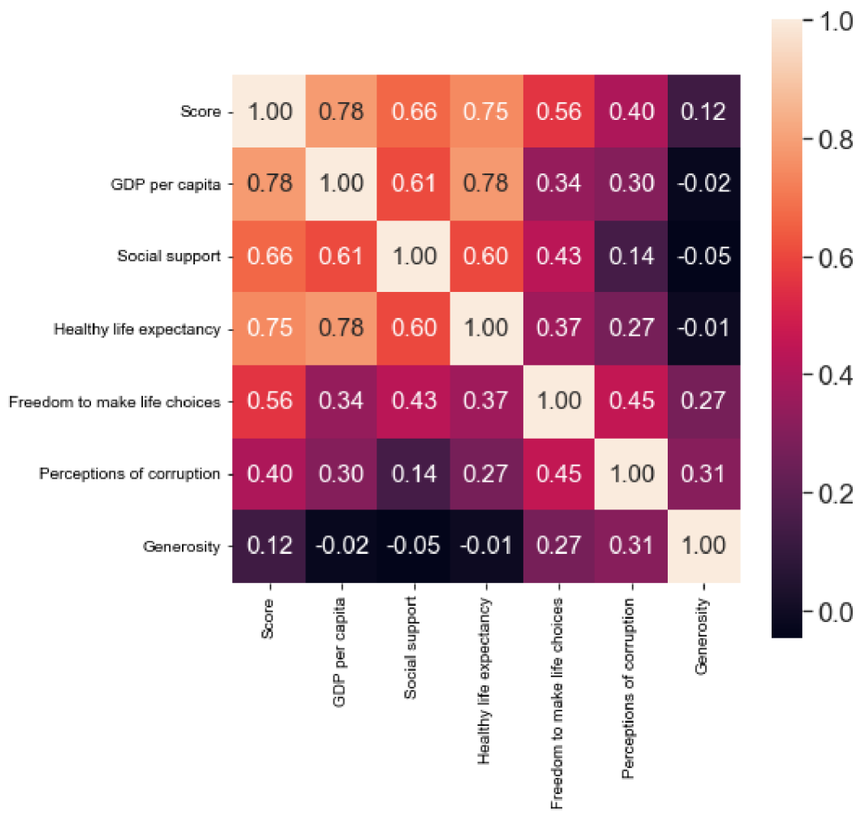
In [23]:
```
# preparing data for Machine Learning - removing unnecessary columns

data_ML = data.copy()

data_ML.reset_index(inplace = True)

del data_ML['Country']
del data_ML['Overall rank']
del data_ML['Year']
data_ML.head(10)
```

Out[23]:

	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Perceptions of corruption	Generosity
0	3.575	0.320	0.303	0.303	0.234	0.097	0.365
1	3.360	0.382	0.110	0.173	0.164	0.071	0.313
2	3.794	0.401	0.582	0.181	0.106	0.061	0.312
3	3.632	0.332	0.537	0.255	0.085	0.036	0.191
4	3.203	0.350	0.517	0.361	0.000	0.025	0.158
5	2.567	0.301	0.356	0.266	0.000	0.001	0.135
6	4.959	0.879	0.804	0.813	0.357	0.064	0.143
7	4.655	0.955	0.502	0.730	0.319	0.053	0.168
8	4.644	0.996	0.804	0.731	0.381	0.040	0.201
9	4.586	0.916	0.817	0.790	0.419	0.032	0.149

In [24]:
```
#  Creating a Heat Map

fig, ax = plt.subplots(figsize = (7,7))
sns.set(font_scale = 1.5)
corr = data_ML.corr()
sns.heatmap(data = corr, square= True, annot = True, cbar = True, fmt = '.2f',
            annot_kws={'size': 15}, xticklabels = data_ML.columns, yticklabels = data_ML.columns)
```

Out[24]: <AxesSubplot:>



SOME MACHINE LEARNING ... :)

In [25]:
```
# importing modules for Machine Learning
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [26]:
```
#  Preparing data and creating model based on most corelated columns, where abs(corelation) > 0.5

mostCorelatedCols = ['GDP per capita', 'Social support', 'Healthy life expectancy','Freedom to make life choices', 'Score']
data_ML = data_ML[mostCorelatedCols]

# Eliminating outliers by IQR method
Q1 = data_ML.quantile(0.25)
Q3 = data_ML.quantile(0.75)
IQR = Q3 - Q1
outlier_condition = (data_ML < Q1 - 1.5 * IQR) | (data_ML > Q3 + 1.5 * IQR)
data_ML_iqr = data_ML[~outlier_condition].dropna()

# spliting data into X - features and y - target
X = data_ML_iqr.drop('Score', axis  = 1)
y = data_ML_iqr['Score'].values

# split the data into learning 80% and testing 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

linear = LinearRegression()
linear.fit(X_train, y_train)

# R2 score for training and testing
r2_train = r2_score(y_train, linear.predict(X_train))
r2_test =  r2_score(y_test, linear.predict(X_test))

#printing results
print('R2 TRAIN ', round(r2_train,3), '\tR2 TEST:', round(r2_test,3))
```

```
R2 TRAIN  0.733         R2 TEST: 0.762
```

In [27]:
```
# Quick look to data

data[mostCorelatedCols].sample(5)
```

Out[27]:

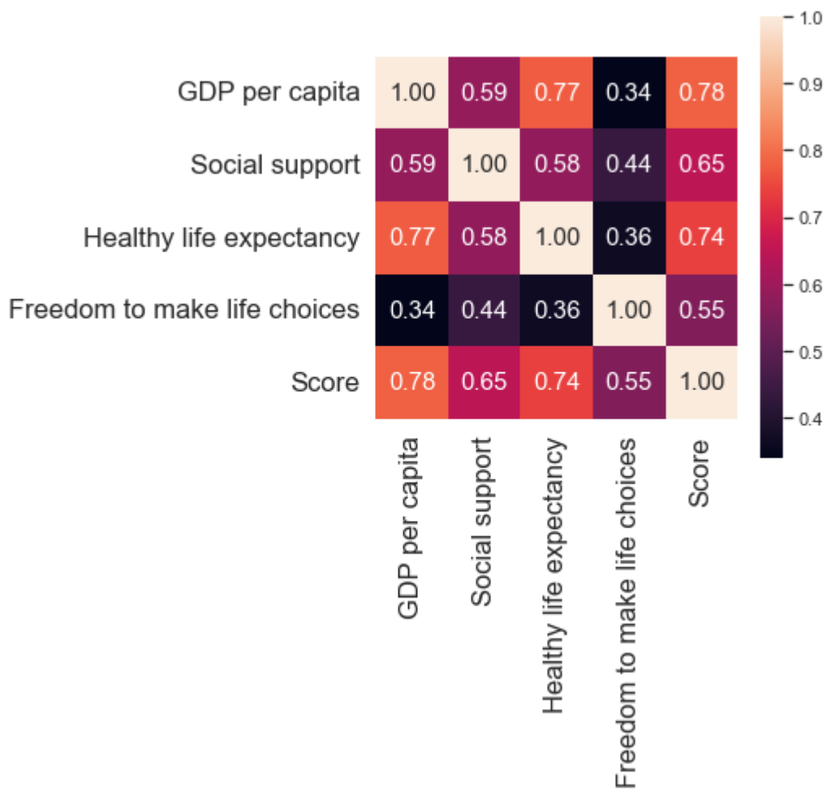
Country	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Score
Bosnia and Herzegovina	0.982	1.069	0.705	0.204	5.182
Brazil	1.088	1.039	0.614	0.404	6.952
India	0.792	0.754	0.455	0.470	4.315
Cameroon	0.525	0.625	0.127	0.427	4.513
Dominican Republic	0.983	1.329	0.742	0.563	5.689

In [28]:
```
#  checking coefficients for trained model:
# "GDP per capita", "Social support", "Healthy life expectancy", "Freedom to make life choices"
linear.coef_
```

Out[28]:
```
array([1.19625716, 0.60010289, 0.92306292, 1.88038698])
```

In [29]:
```
#  checking the "heat map" for data taken for machine learning

fig, ax = plt.subplots(figsize = (5,5))
sns.set(font_scale = 1)
corr = data_ML_iqr.corr()
sns.heatmap(data = corr, square= True, annot = True, cbar = True, fmt = '.2f',
            annot_kws={'size': 15}, xticklabels = data_ML_iqr.columns, yticklabels = data_ML_iqr.columns)
```

Out[29]: <AxesSubplot:>



Conclusion: The correlation coefficients between data features (heat map) are moderately related to the resulting model coefficients (model.coef_)

In [30]:
```
# Making a prediction for sample countries (from 2020, becouse data was sorted ascending, and we use tail method)

sampleCountries = ['Poland', 'Germany', 'Russia', 'Mexico', 'Australia', 'Lesotho', 'Brazil', 'Norway', 'Ukraine']
predictedValues = []
for country in sampleCountries:

    value_1 = float(data[mostCorelatedCols].loc[country,'GDP per capita'].tail(1))
    value_2 = float(data[mostCorelatedCols].loc[country,'Social support'].tail(1))
    value_3 = float(data[mostCorelatedCols].loc[country,'Healthy life expectancy'].tail(1))
    value_4 = float(data[mostCorelatedCols].loc[country,'Freedom to make life choices'].tail(1))

    prediction = linear.predict([[value_1, value_2, value_3, value_4]])
    prediction = round(prediction[0], 3)
    predictedValues.append(prediction)
predictedValues
```

Out[30]:
```
[6.306, 6.622, 5.824, 6.041, 6.838, 4.331, 5.844, 7.061, 5.241]
```

In [31]:
```
# Checking real "Score" values for sample countries

realValues = []
for country in sampleCountries:
    real_value = float(data[mostCorelatedCols].loc[country, 'Score'].tail(1))
    realValues.append(real_value)
realValues
```

Out[31]:
```
[6.186, 7.076, 5.546, 6.465, 7.223, 3.653, 6.376, 7.488, 4.561]
```

In [32]:
```
# comparing predictions with real "Score" Values

for predict, real, country in zip(predictedValues, realValues, sampleCountries):
    print('For {0:^12s} real "Score" value was: {1:^8.2f}| model predicts: {2:^8.2f}| absolute error is {3:^8.2f}'.format(country, real, predict, real - predict))
    print('-' * 100)
```

```
For    Poland    real "Score" value was:   6.19  | model predicts:   6.31  | absolute error is  -0.12
----------------------------------------------------------------------------------------------------
For   Germany    real "Score" value was:   7.08  | model predicts:   6.62  | absolute error is   0.45
----------------------------------------------------------------------------------------------------
For    Russia    real "Score" value was:   5.55  | model predicts:   5.82  | absolute error is  -0.28
----------------------------------------------------------------------------------------------------
For    Mexico    real "Score" value was:   6.46  | model predicts:   6.04  | absolute error is   0.42
----------------------------------------------------------------------------------------------------
For  Australia   real "Score" value was:   7.22  | model predicts:   6.84  | absolute error is   0.38
----------------------------------------------------------------------------------------------------
For   Lesotho    real "Score" value was:   3.65  | model predicts:   4.33  | absolute error is  -0.68
----------------------------------------------------------------------------------------------------
For    Brazil    real "Score" value was:   6.38  | model predicts:   5.84  | absolute error is   0.53
----------------------------------------------------------------------------------------------------
For    Norway    real "Score" value was:   7.49  | model predicts:   7.06  | absolute error is   0.43
----------------------------------------------------------------------------------------------------
For   Ukraine    real "Score" value was:   4.56  | model predicts:   5.24  | absolute error is  -0.68
----------------------------------------------------------------------------------------------------
```

In [33]:
```
# Making a prediction for luxury "Wonderland" with very high factors

GBP = 2.0
socialSupport = 1.6
lifeExpectancy = 1.6
freedom = 1.5

prediction = linear.predict([[GBP, socialSupport, lifeExpectancy, freedom]])
prediction = round(prediction[0], 3)

print('Happyness factor for "Wonderland" is equal to', prediction)
```

```
Happyness factor for "Wonderland" is equal to 9.921
```