In [44]:

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

In [45]:

```
data = pd.read_csv("happiness_rankings.csv")
```

In [46]:

```
data
```

Out[46]:

	RANK	Country	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explai by: Hea expecta
0	1	Finland	7.821	7.886	7.756	2.518	1.892	1.258	0.
1	2	Denmark	7.636	7.710	7.563	2.226	1.953	1.243	0.
2	3	Iceland	7.557	7.651	7.464	2.320	1.936	1.320	0.
3	4	Switzerland	7.512	7.586	7.437	2.153	2.026	1.226	0.
4	5	Netherlands	7.415	7.471	7.359	2.137	1.945	1.206	0.
...	...	...	...	...	...	...	...	...	
141	142	Botswana	3.471	3.667	3.275	0.187	1.503	0.815	0.
142	143	Rwanda	3.268	3.462	3.074	0.536	0.785	0.133	0.
143	144	Zimbabwe	2.995	3.110	2.880	0.548	0.947	0.690	0.
144	145	Lebanon	2.955	3.049	2.862	0.216	1.392	0.498	0.
145	146	Afghanistan	2.404	2.469	2.339	1.263	0.758	0.000	0.

146 rows × 12 columns

In [47]:

```
data.head()
```

Out[47]:

	RANK	Country	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explaine by: Health lif expectanc
0	1	Finland	7.821	7.886	7.756	2.518	1.892	1.258	0.77
1	2	Denmark	7.636	7.710	7.563	2.226	1.953	1.243	0.77
2	3	Iceland	7.557	7.651	7.464	2.320	1.936	1.320	0.80
3	4	Switzerland	7.512	7.586	7.437	2.153	2.026	1.226	0.82
4	5	Netherlands	7.415	7.471	7.359	2.137	1.945	1.206	0.78

In [48]:

```
data.tail()
```

Out[48]:

	RANK	Country	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explai by: Hea expecta
141	142	Botswana	3.471	3.667	3.275	0.187	1.503	0.815	0.:
142	143	Rwanda	3.268	3.462	3.074	0.536	0.785	0.133	0.‹
143	144	Zimbabwe	2.995	3.110	2.880	0.548	0.947	0.690	0.:
144	145	Lebanon	2.955	3.049	2.862	0.216	1.392	0.498	0.(
145	146	Afghanistan	2.404	2.469	2.339	1.263	0.758	0.000	0.:

In [49]:

```
data.shape
```

Out[49]:

```
(146, 12)
```

In [50]:

```
data.columns
```

Out[50]:

```
Index(['RANK', 'Country', 'Happiness score', 'Whisker-high', 'Whisker-lo
w',
       'Dystopia (1.83) + residual', 'Explained by: GDP per capita',
       'Explained by: Social support', 'Explained by: Healthy life expecta
ncy',
       'Explained by: Freedom to make life choices',
       'Explained by: Generosity', 'Explained by: Perceptions of corruptio
n'],
      dtype='object')
```

In [51]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146 entries, 0 to 145
Data columns (total 12 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   RANK                                        146 non-null    int64  
 1   Country                                     146 non-null    object 
 2   Happiness score                             146 non-null    float64
 3   Whisker-high                                146 non-null    float64
 4   Whisker-low                                 146 non-null    float64
 5   Dystopia (1.83) + residual                  146 non-null    float64
 6   Explained by: GDP per capita                146 non-null    float64
 7   Explained by: Social support                146 non-null    float64
 8   Explained by: Healthy life expectancy       146 non-null    float64
 9   Explained by: Freedom to make life choices  146 non-null    float64
 10  Explained by: Generosity                    146 non-null    float64
 11  Explained by: Perceptions of corruption     146 non-null    float64
dtypes: float64(10), int64(1), object(1)
memory usage: 13.8+ KB
```

In [52]:

```
data.describe()
```

Out[52]:

	RANK	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	e
count	146.000000	146.000000	146.000000	146.000000	146.000000	146.000000	146.000000	1
mean	73.500000	5.553575	5.673589	5.433568	1.831808	1.410445	0.905863	
std	42.290661	1.086843	1.065621	1.109380	0.534994	0.421663	0.280122	
min	1.000000	2.404000	2.469000	2.339000	0.187000	0.000000	0.000000	
25%	37.250000	4.888750	5.006250	4.754750	1.555250	1.095500	0.732000	
50%	73.500000	5.568500	5.680000	5.453000	1.894500	1.445500	0.957500	
75%	109.750000	6.305000	6.448750	6.190000	2.153000	1.784750	1.114250	
max	146.000000	7.821000	7.886000	7.756000	2.844000	2.209000	1.320000	

In [53]:

```
data.isnull().sum()
```

Out[53]:

```
RANK                                          0
Country                                       0
Happiness score                               0
Whisker-high                                  0
Whisker-low                                   0
Dystopia (1.83) + residual                    0
Explained by: GDP per capita                  0
Explained by: Social support                  0
Explained by: Healthy life expectancy         0
Explained by: Freedom to make life choices    0
Explained by: Generosity                      0
Explained by: Perceptions of corruption       0
dtype: int64
```

In [54]:

```
data_country = data.groupby('Country').sum()
data_country.sort_values(by = 'Happiness score', ascending = False)
```

Out[54]:

Country	RANK	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explaine by: Health lif expectanc
Cyprus	120	11.688	11.929	11.447	3.122	3.630	1.797	1.63
Finland	1	7.821	7.886	7.756	2.518	1.892	1.258	0.77
Denmark	2	7.636	7.710	7.563	2.226	1.953	1.243	0.77
Iceland	3	7.557	7.651	7.464	2.320	1.936	1.320	0.80
Switzerland	4	7.512	7.586	7.437	2.153	2.026	1.226	0.82
...	...	...	...	...	...	...	...	.
Botswana	142	3.471	3.667	3.275	0.187	1.503	0.815	0.28
Rwanda	143	3.268	3.462	3.074	0.536	0.785	0.133	0.46
Zimbabwe	144	2.995	3.110	2.880	0.548	0.947	0.690	0.27
Lebanon	145	2.955	3.049	2.862	0.216	1.392	0.498	0.63
Afghanistan	146	2.404	2.469	2.339	1.263	0.758	0.000	0.28

145 rows × 11 columns

In [55]:

```
data_country.sort_values(by = 'Happiness score', ascending = False).head()
```

Out[55]:

Country	RANK	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explained by: Healthy life expectancy
Cyprus	120	11.688	11.929	11.447	3.122	3.630	1.797	1.638
Finland	1	7.821	7.886	7.756	2.518	1.892	1.258	0.775
Denmark	2	7.636	7.710	7.563	2.226	1.953	1.243	0.777
Iceland	3	7.557	7.651	7.464	2.320	1.936	1.320	0.803
Switzerland	4	7.512	7.586	7.437	2.153	2.026	1.226	0.822

In [56]:

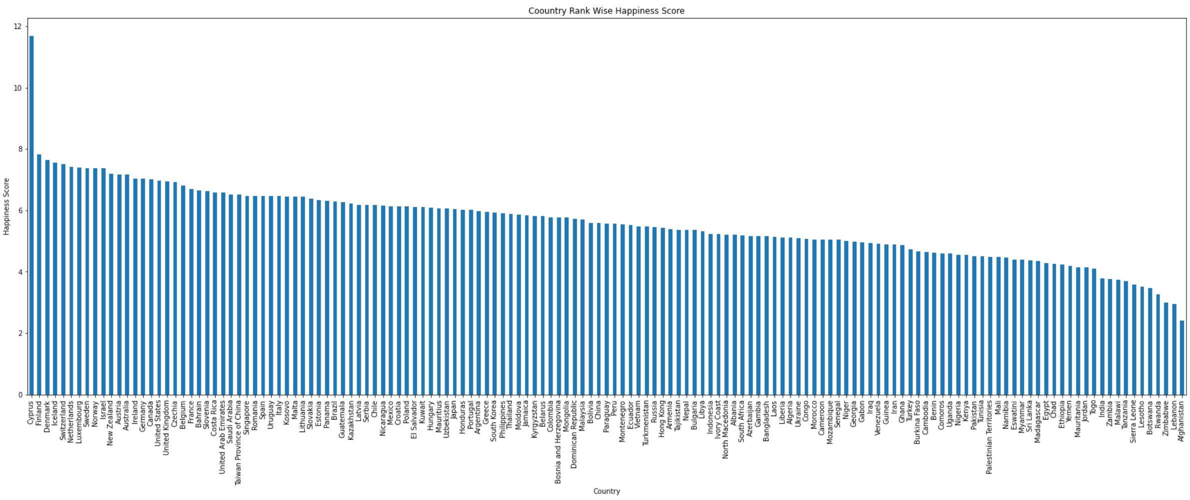
```
data_country.sort_values(by = 'Happiness score', ascending = False).tail()
```

Out[56]:

Country	RANK	Happiness score	Whisker-high	Whisker-low	Dystopia (1.83) + residual	Explained by: GDP per capita	Explained by: Social support	Explained by: Healthy life expectancy
Botswana	142	3.471	3.667	3.275	0.187	1.503	0.815	0.280
Rwanda	143	3.268	3.462	3.074	0.536	0.785	0.133	0.462
Zimbabwe	144	2.995	3.110	2.880	0.548	0.947	0.690	0.270
Lebanon	145	2.955	3.049	2.862	0.216	1.392	0.498	0.631
Afghanistan	146	2.404	2.469	2.339	1.263	0.758	0.000	0.289

In [57]:

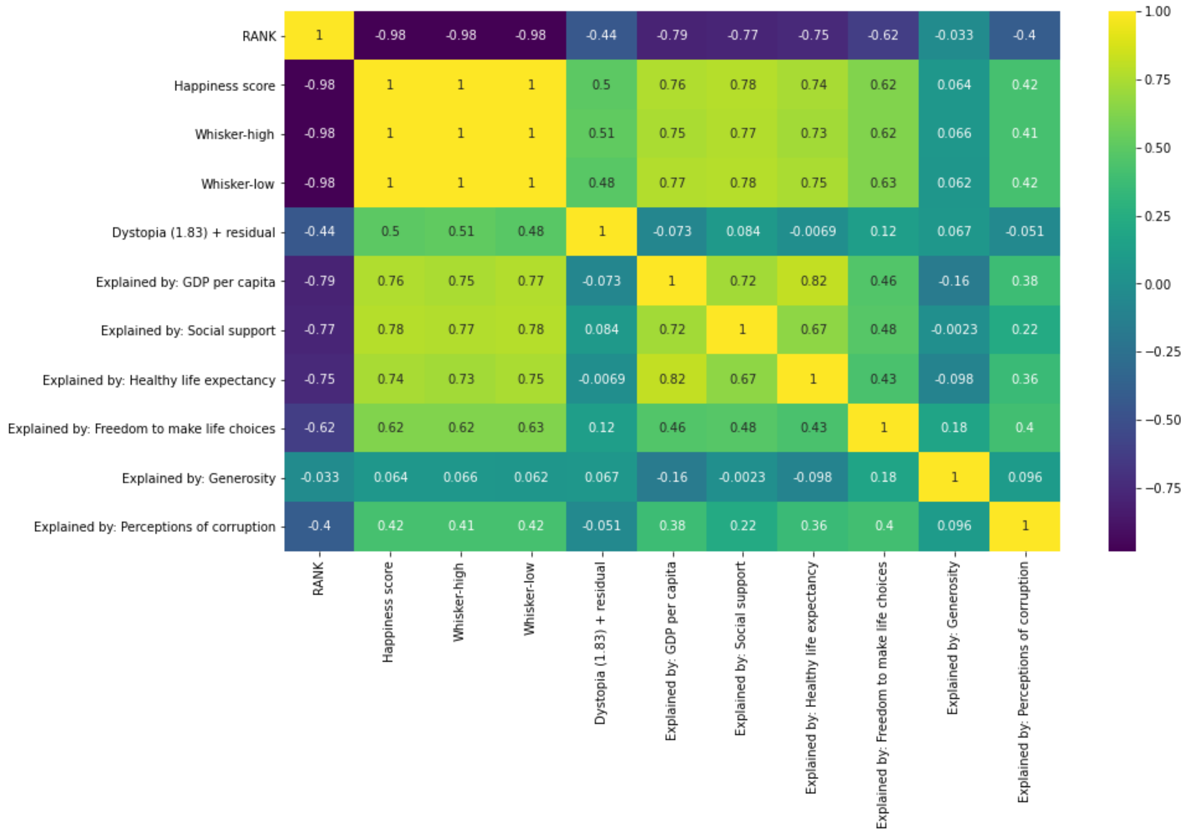
```
plt.subplots(figsize = (30, 10))
cr = data_country['Happiness score'].sort_values(ascending = False)
ax = cr.plot.bar()
ax.set_xlabel('Country')
ax.set_ylabel('Happiness Score')
ax.set_title('Coountry Rank Wise Happiness Score')
plt.show()
print(cr)
```




```
Country
Cyprus         11.688
Finland         7.821
Denmark         7.636
Iceland         7.557
Switzerland     7.512
                ...
Botswana        3.471
Rwanda          3.268
Zimbabwe        2.995
Lebanon         2.955
Afghanistan     2.404
Name: Happiness score, Length: 145, dtype: float64
```

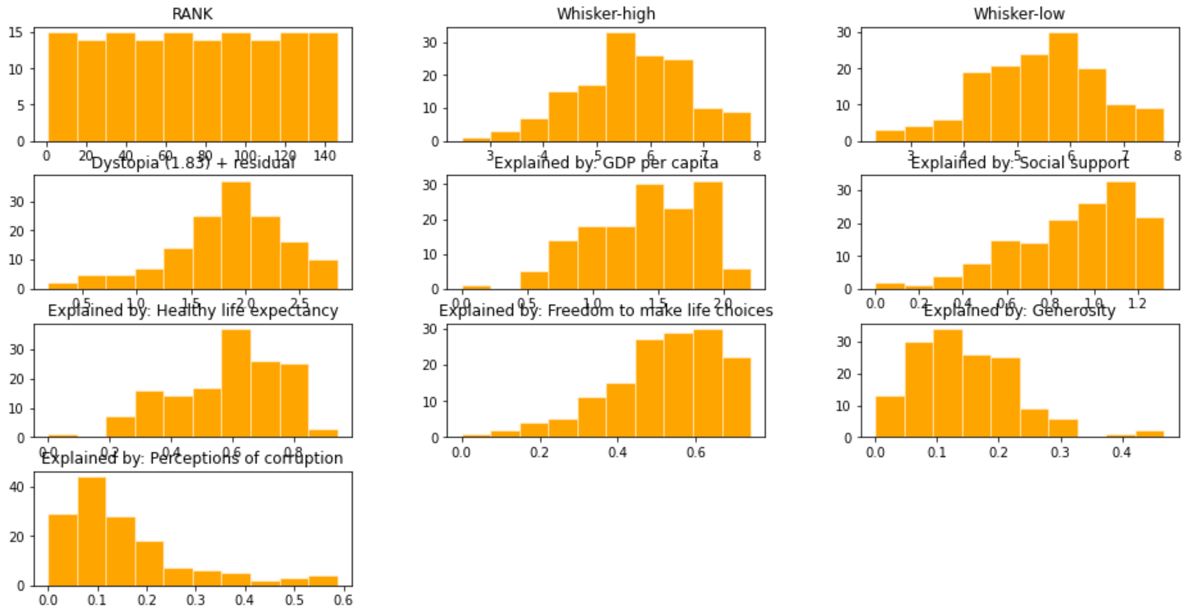
In [58]:

```
fig, ax = plt.subplots(figsize=(14, 8))
sns.heatmap(data.corr(), annot=True, cmap="viridis");
```



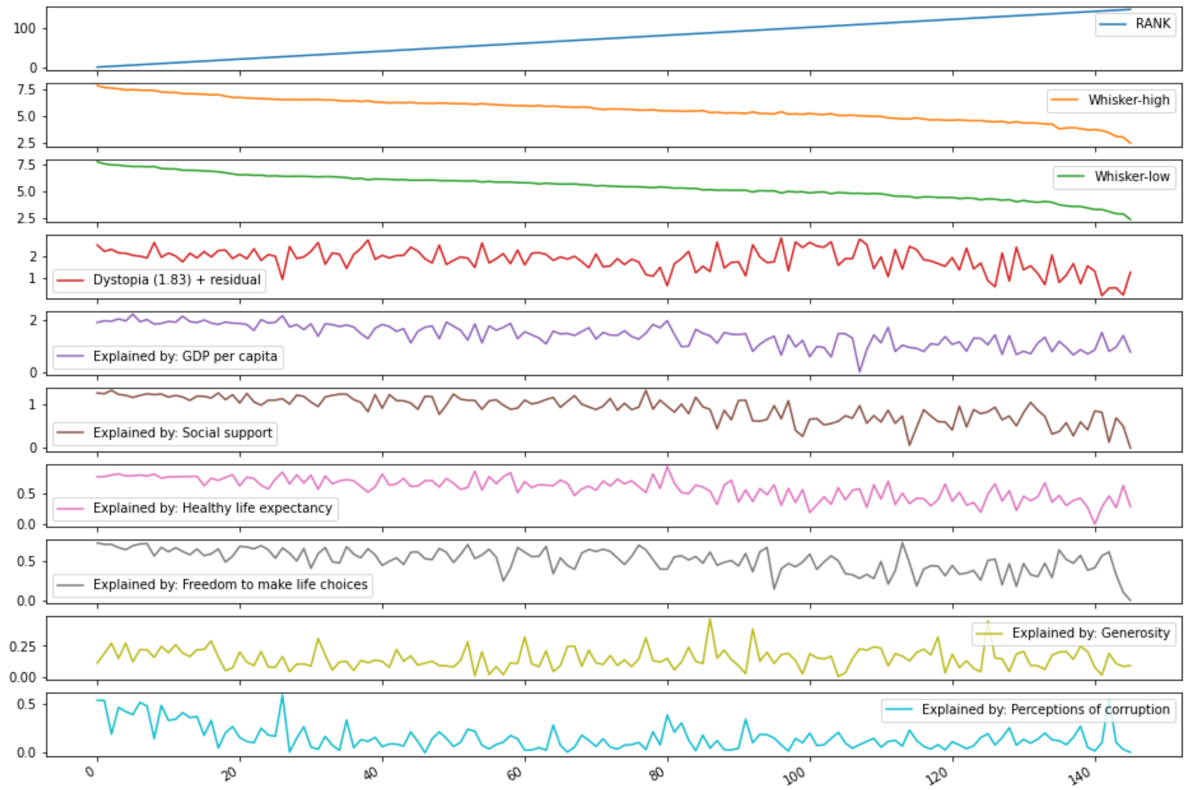
In [59]:

```
data.drop(['Happiness score'], axis = 1).hist(edgecolor = 'white',
                                        linewidth = 0.5,
                                        figsize = (16,8),grid=False,
                                        color='orange')
plt.show()
```



In [60]:

```
data.drop('Happiness score',axis=1).plot(subplots=True, figsize=(16, 12));
```



In [81]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [82]:

```
data.Country = le.fit_transform(data.Country)
```

In [83]:

```
x = data.drop(['RANK'], axis=1)
y = data["RANK"]
```

In [84]:

```
from sklearn.preprocessing import StandardScaler
```

In [85]:

```
scale = StandardScaler()
sdata = scale.fit_transform(x.drop(['Country'], axis=1))
```

In [86]:

```
from sklearn.linear_model import LinearRegression
```

In [87]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [88]:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[88]:

```
LinearRegression()
```

In [90]:

```
y_pred = model.predict(X_test)
```

In [92]:

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.9661844826452385
Testing Accuracy : 0.9716196480566516
```

In [93]:

```
model.get_params(deep = True)
```

Out[93]:

```
{'copy_X': True,
 'fit_intercept': True,
 'n_jobs': None,
 'normalize': 'deprecated',
 'positive': False}
```

In [94]:

```
model1 = model.fit(X_train, y_train)
pred = model.predict(X_test)
model.score(X_test,y_test)
```

Out[94]:

```
0.9716196480566516
```