In [118]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [119]:
```
y_2015 = pd.read_csv('Happiness/2015.csv');
y_2016 = pd.read_csv('Happiness/2016.csv');
y_2017 = pd.read_csv('Happiness/2017.csv');
y_2018 = pd.read_csv('Happiness/2018.csv');
y_2019 = pd.read_csv('Happiness/2019.csv');
```

In [120]:
```
y_2015['Year']=2015
y_2015.columns = ['Country', 'Region','Happiness_Rank', 'Happiness_Score', 'St
       'GDP_per_Capita', 'Social', 'Healthy_life_expectancy', 'Freedom',
       'Corruption', 'Generosity', 'Dystopia_Residual','Year']
```

In [121]:
```
y_2016['Year']=2016
y_2016.columns = ['Country','Region', 'Happiness_Rank', 'Happiness_Score',
       'Lower Confidence Interval', 'Upper Confidence Interval',
       'GDP_per_Capita', 'Social', 'Healthy_life_expectancy',
       'Freedom', 'Corruption', 'Generosity',
       'Dystopia_Residual','Year']
```

In [122]:
```
y_2017['Year']=2017
y_2017.columns = ['Country', 'Happiness_Rank', 'Happiness_Score', 'Whisker_hig
       'Whisker_low', 'GDP_per_Capita', 'Social',
       'Healthy_life_expectancy', 'Freedom', 'Generosity',
       'Corruption', 'Dystopia_Residual','Year']
```

In [123]:
```
y_2018['Year']=2018
y_2018.columns = ['Happiness_Rank', 'Country', 'Happiness_Score', 'GDP_per_Cap
                  'Freedom', 'Generosity', 'Corruption','Year']
```

In [124]:
```
y_2019['Year']=2019
y_2019.columns = ['Happiness_Rank', 'Country', 'Happiness_Score', 'GDP_per_Cap
                  'Freedom', 'Generosity', 'Corruption','Year']
```

In [125]:
```
y_2015.head()
```

Out[125]:

	Country	Region	Happiness_Rank	Happiness_Score	Standard Error	GDP_per_Capita	Social
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261

In [126]:
```
y_2016.head()
```

Out[126]:

	Country	Region	Happiness_Rank	Happiness_Score	Lower Confidence Interval	Upper Confidence Interval	GDP_per_C
0	Denmark	Western Europe	1	7.526	7.460	7.592	1.4
1	Switzerland	Western Europe	2	7.509	7.428	7.590	1.5
2	Iceland	Western Europe	3	7.501	7.333	7.669	1.4
3	Norway	Western Europe	4	7.498	7.421	7.575	1.5
4	Finland	Western Europe	5	7.413	7.351	7.475	1.4

In [127]:
```
y_2017.head()
```

Out[127]:

	Country	Happiness_Rank	Happiness_Score	Whisker_high	Whisker_low	GDP_per_Capita
0	Norway	1	7.537	7.594445	7.479556	1.616463
1	Denmark	2	7.522	7.581728	7.462272	1.482383
2	Iceland	3	7.504	7.622030	7.385970	1.480633
3	Switzerland	4	7.494	7.561772	7.426227	1.564980
4	Finland	5	7.469	7.527542	7.410458	1.443572

In [128]:
```
y_2018.head()
```

Out[128]:

	Happiness_Rank	Country	Happiness_Score	GDP_per_Capita	Social	Healthy_life_expectan
0	1	Finland	7.632	1.305	1.592	0.8
1	2	Norway	7.594	1.456	1.582	0.8
2	3	Denmark	7.555	1.351	1.590	0.8
3	4	Iceland	7.495	1.343	1.644	0.9
4	5	Switzerland	7.487	1.420	1.549	0.9

In [129]:
```
y_2019.head()
```

Out[129]:

	Happiness_Rank	Country	Happiness_Score	GDP_per_Capita	Social	Healthy_life_expectar
0	1	Finland	7.769	1.340	1.587	0.9
1	2	Denmark	7.600	1.383	1.573	0.9
2	3	Norway	7.554	1.488	1.582	1.0
3	4	Iceland	7.494	1.380	1.624	1.0
4	5	Netherlands	7.488	1.396	1.522	0.9

In [130]:
```
Alldata = pd.concat([y_2015,y_2016,y_2017,y_2018,y_2019])
Alldata
```

Out[130]:

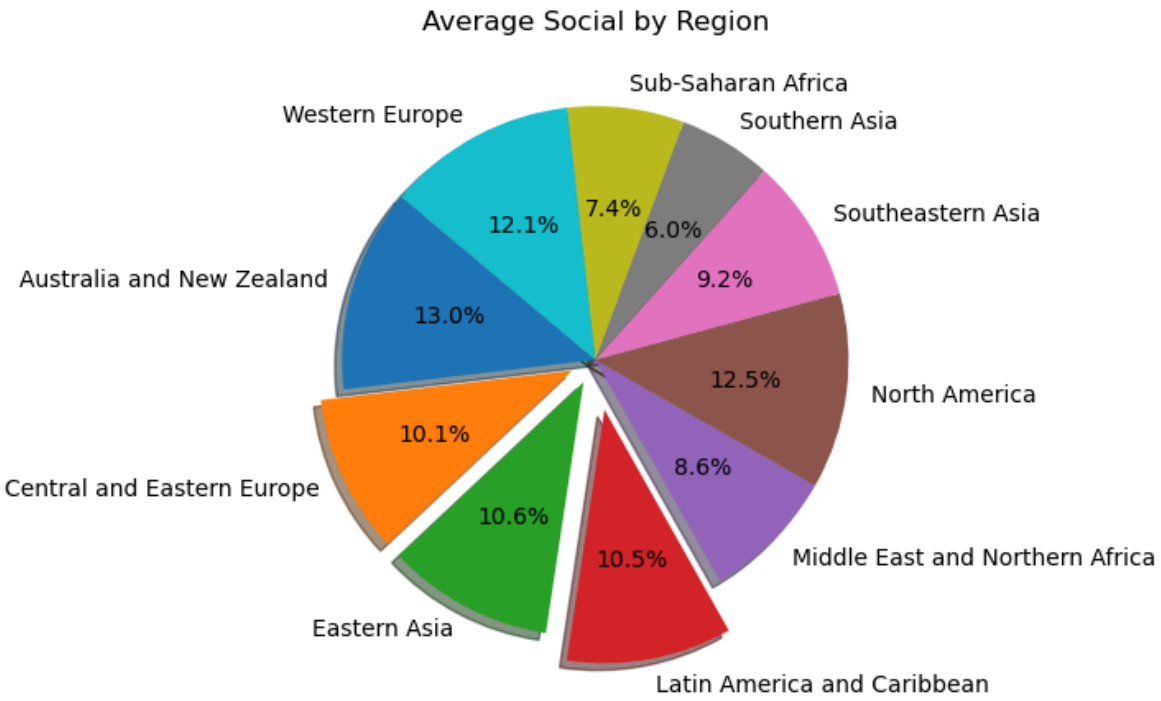
	Country	Region	Happiness_Rank	Happiness_Score	Standard Error	GDP_per_Capita	Socia
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.3495
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.4022
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.3605
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.3309
4	Canada	North America	5	7.427	0.03553	1.32629	1.3226
...	...	...	...	...	...	...	.
151	Rwanda	NaN	152	3.334	NaN	0.35900	0.7110
152	Tanzania	NaN	153	3.231	NaN	0.47600	0.8850
153	Afghanistan	NaN	154	3.203	NaN	0.35000	0.5170
154	Central African Republic	NaN	155	3.083	NaN	0.02600	0.0000
155	South Sudan	NaN	156	2.853	NaN	0.30600	0.5750

782 rows × 17 columns

In [131]:
```
from sklearn.cluster import KMeans
```

In [132]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [136]:
```
social_by_region = Alldata.groupby('Region')['Social'].mean()
plt.figure(figsize=(5, 5))
plt.pie(social_by_region, labels=social_by_region.index, autopct='%1.1f%%',
        startangle=140,shadow=True,explode=[0,0.1,0.1,0.2,0,0,0,0,0,0])
plt.title('Average Social by Region')
plt.show()
```



In [137]:
```
Data = Alldata.groupby('Region')['Happiness_Score'].mean()
Data
```

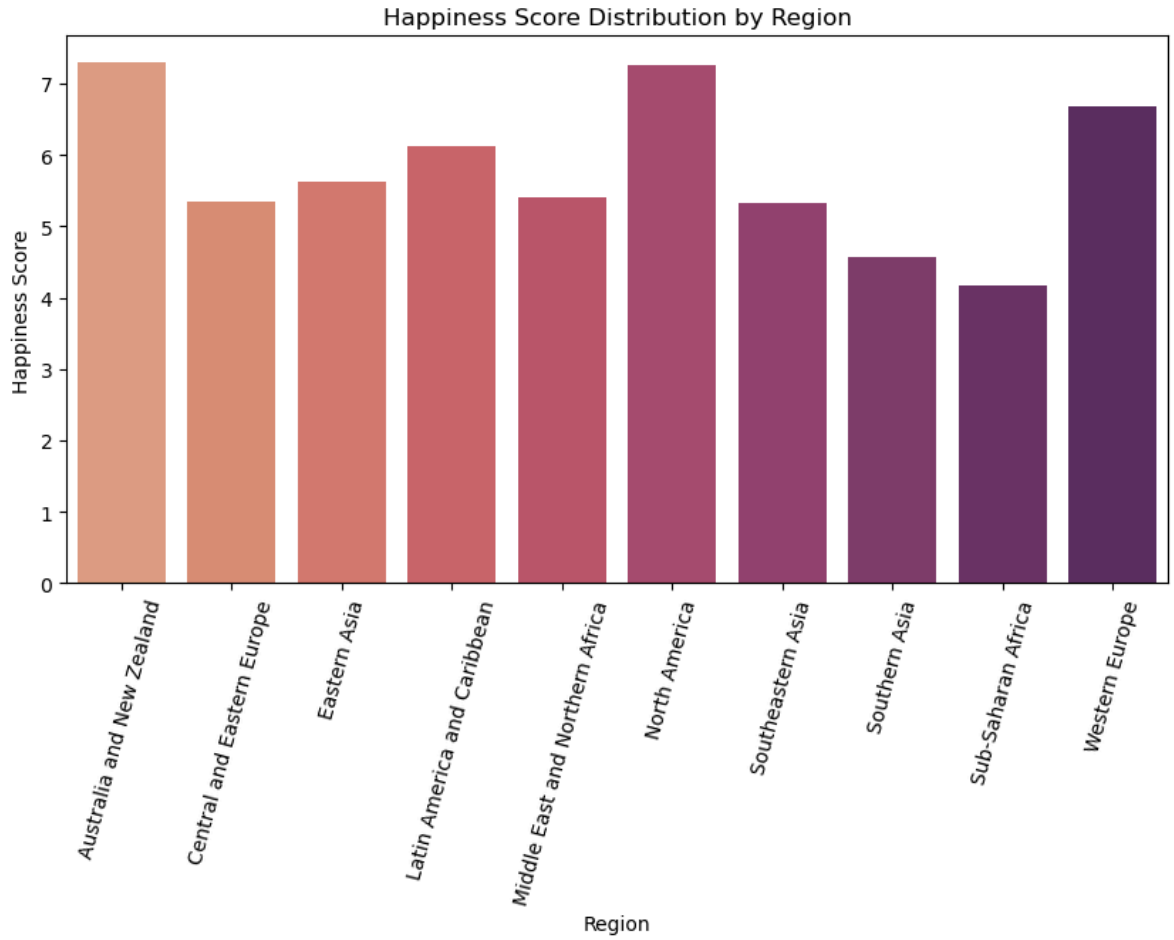
Out[137]:
```
Region
Australia and New Zealand          7.304250
Central and Eastern Europe         5.351810
Eastern Asia                       5.625167
Latin America and Caribbean        6.122283
Middle East and Northern Africa    5.396744
North America                      7.263500
Southeastern Asia                  5.328167
Southern Asia                      4.572071
Sub-Saharan Africa                 4.170462
Western Europe                     6.687643
Name: Happiness_Score, dtype: float64
```

In [138]:
```
Data = Alldata.groupby('Region')['Happiness_Score'].mean().reset_index()
Data
```

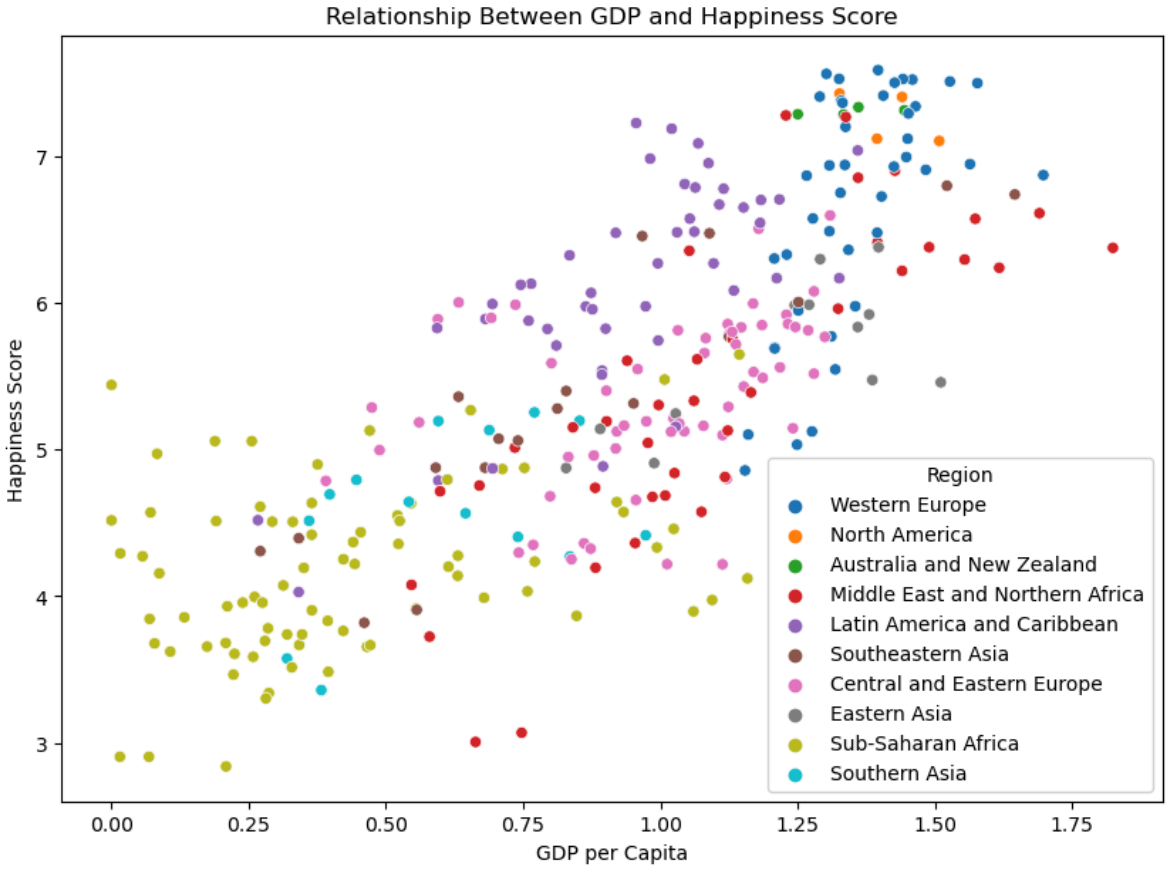
Out[138]:

	Region	Happiness_Score
0	Australia and New Zealand	7.304250
1	Central and Eastern Europe	5.351810
2	Eastern Asia	5.625167
3	Latin America and Caribbean	6.122283
4	Middle East and Northern Africa	5.396744
5	North America	7.263500
6	Southeastern Asia	5.328167
7	Southern Asia	4.572071
8	Sub-Saharan Africa	4.170462
9	Western Europe	6.687643

In [139]:
```
plt.figure(figsize=(10, 5))
sns.barplot(x="Region",y="Happiness_Score",data=Data,palette='flare')
plt.xlabel('Region')
plt.ylabel('Happiness Score')
plt.title('Happiness Score Distribution by Region')
plt.xticks(rotation = 75 )
plt.show()
```



In [140]:
```
plt.figure(figsize=[10,7])
sns.scatterplot(
    x="GDP_per_Capita",
    y="Happiness_Score",
    hue="Region",
    data=Alldata)
plt.xlabel('GDP per Capita')
plt.ylabel('Happiness Score')
plt.title('Relationship Between GDP and Happiness Score')
plt.show()
```



In [141]:
```
Alldata['Region'].unique()
```

Out[141]:
```
array(['Western Europe', 'North America', 'Australia and New Zealand',
       'Middle East and Northern Africa', 'Latin America and Caribbean',
       'Southeastern Asia', 'Central and Eastern Europe', 'Eastern Asia',
       'Sub-Saharan Africa', 'Southern Asia', nan], dtype=object)
```

In [142]:
```
Alldata['Region'].nunique()
```

Out[142]:
```
10
```

In [143]:
```
Alldata['Country'].unique()
```

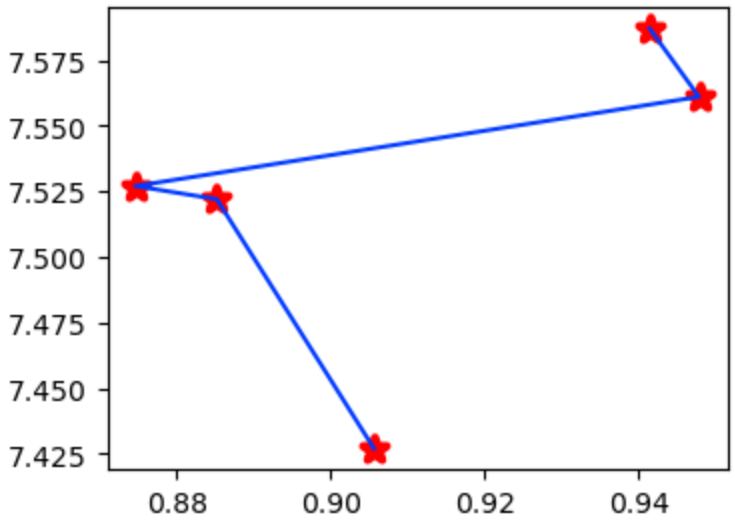
Out[143]:
```
array(['Switzerland', 'Iceland', 'Denmark', 'Norway', 'Canada', 'Finland',
       'Netherlands', 'Sweden', 'New Zealand', 'Australia', 'Israel',
       'Costa Rica', 'Austria', 'Mexico', 'United States', 'Brazil',
       'Luxembourg', 'Ireland', 'Belgium', 'United Arab Emirates',
       'United Kingdom', 'Oman', 'Venezuela', 'Singapore', 'Panama',
       'Germany', 'Chile', 'Qatar', 'France', 'Argentina',
       'Czech Republic', 'Uruguay', 'Colombia', 'Thailand',
       'Saudi Arabia', 'Spain', 'Malta', 'Taiwan', 'Kuwait', 'Suriname',
       'Trinidad and Tobago', 'El Salvador', 'Guatemala', 'Uzbekistan',
       'Slovakia', 'Japan', 'South Korea', 'Ecuador', 'Bahrain', 'Italy',
       'Bolivia', 'Moldova', 'Paraguay', 'Kazakhstan', 'Slovenia',
       'Lithuania', 'Nicaragua', 'Peru', 'Belarus', 'Poland', 'Malaysia',
       'Croatia', 'Libya', 'Russia', 'Jamaica', 'North Cyprus', 'Cyprus',
       'Algeria', 'Kosovo', 'Turkmenistan', 'Mauritius', 'Hong Kong',
       'Estonia', 'Indonesia', 'Vietnam', 'Turkey', 'Kyrgyzstan',
       'Nigeria', 'Bhutan', 'Azerbaijan', 'Pakistan', 'Jordan',
       'Montenegro', 'China', 'Zambia', 'Romania', 'Serbia', 'Portugal',
       'Latvia', 'Philippines', 'Somaliland region', 'Morocco',
       'Macedonia', 'Mozambique', 'Albania', 'Bosnia and Herzegovina',
       'Lesotho', 'Dominican Republic', 'Laos', 'Mongolia', 'Swaziland',
       'Greece', 'Lebanon', 'Hungary', 'Honduras', 'Tajikistan',
       'Tunisia', 'Palestinian Territories', 'Bangladesh', 'Iran',
       'Ukraine', 'Iraq', 'South Africa', 'Ghana', 'Zimbabwe', 'Liberia',
       'India', 'Sudan', 'Haiti', 'Congo (Kinshasa)', 'Nepal', 'Ethiopia',
       'Sierra Leone', 'Mauritania', 'Kenya', 'Djibouti', 'Armenia',
       'Botswana', 'Myanmar', 'Georgia', 'Malawi', 'Sri Lanka',
       'Cameroon', 'Bulgaria', 'Egypt', 'Yemen', 'Angola', 'Mali',
       'Congo (Brazzaville)', 'Comoros', 'Uganda', 'Senegal', 'Gabon',
       'Niger', 'Cambodia', 'Tanzania', 'Madagascar',
       'Central African Republic', 'Chad', 'Guinea', 'Ivory Coast',
       'Burkina Faso', 'Afghanistan', 'Rwanda', 'Benin', 'Syria',
       'Burundi', 'Togo', 'Puerto Rico', 'Belize', 'Somalia',
       'Somaliland Region', 'Namibia', 'South Sudan',
       'Taiwan Province of China', 'Hong Kong S.A.R., China',
       'Trinidad & Tobago', 'Northern Cyprus', 'North Macedonia',
       'Gambia'], dtype=object)
```

In [144]:
```
Alldata['Country'].nunique()
```

Out[144]:
```
170
```

In [145]:
```
att = Alldata[['Healthy_life_expectancy']].head()
label = Alldata['Happiness_Score'].head()

from sklearn.preprocessing import PolynomialFeatures
pf = PolynomialFeatures(degree=4)
att_new = pf.fit(att).transform(att)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(att_new , label)

plt.figure(figsize=[4,3])
plt.scatter(att['Healthy_life_expectancy'],label,marker='*',s=80,lw=3,color="#
plt.plot(att['Healthy_life_expectancy'],model.predict(att_new),'#003AFF')

print('Accuracy',model.score(att_new,label))
```

```
Accuracy 1.0
```



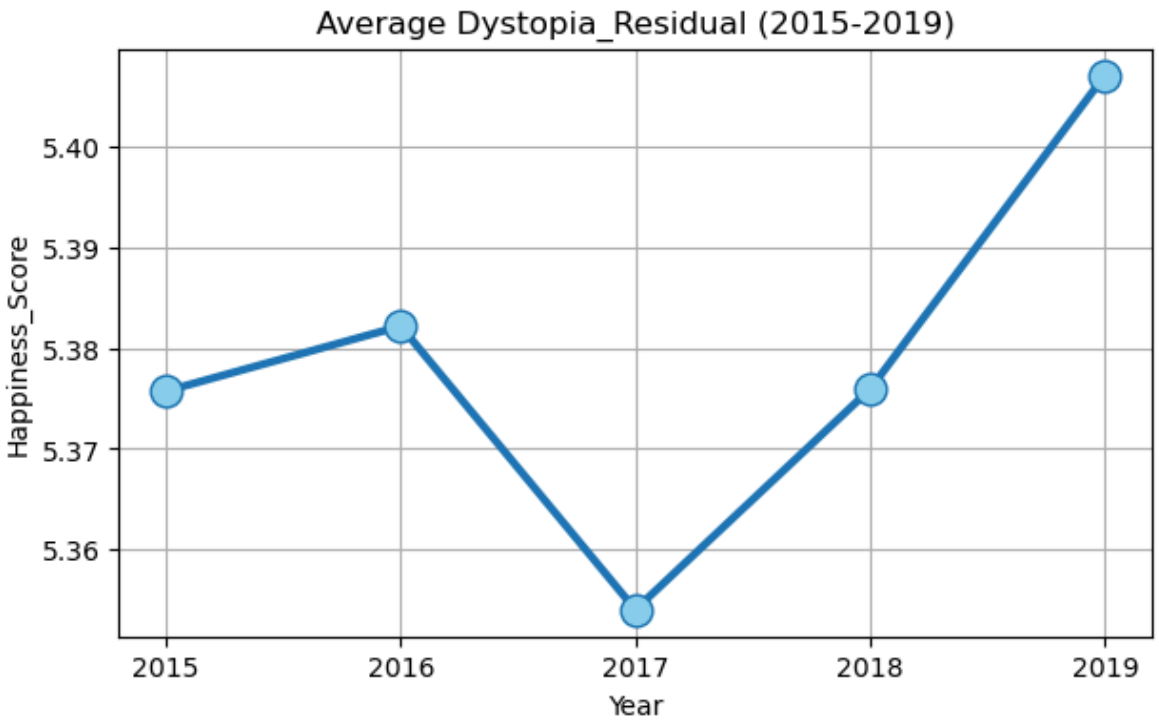
In [146]:
```
data = {
    "Year": [2015, 2016, 2017,2018,2019],
    "Happiness_Score": [y_2015["Happiness_Score"].mean(), y_2016["Happiness_Sc
                        y_2017["Happiness_Score"].mean(),y_2018["Happiness_Scc
                        y_2019["Happiness_Score"].mean()]}
```

In [147]:
```
data
```

Out[147]:
```
{'Year': [2015, 2016, 2017, 2018, 2019],
 'Happiness_Score': [5.375734177215189,
  5.382184713375795,
  5.354019355773926,
  5.375916666666667,
  5.407096153846155]}
```

In [148]:
```
df = pd.DataFrame(data)
```

In [149]:
```
plt.figure(figsize=(7, 4))
plt.plot(df["Year"], df["Happiness_Score"], marker='o', markerfacecolor='skybl
         , linewidth=3)
plt.title('Average Dystopia_Residual (2015-2019)')
plt.xlabel('Year')
plt.ylabel('Happiness_Score')
plt.xticks([2015, 2016,2017,2018,2019])
plt.grid(True)
plt.show()
```



In [150]:
```
T_15 = y_2015[y_2015["Country"]=="Thailand"]
T_16 = y_2016[y_2016["Country"]=="Thailand"]
T_17 = y_2017[y_2017["Country"]=="Thailand"]
T_18 = y_2018[y_2018["Country"]=="Thailand"]
T_19 = y_2019[y_2019["Country"]=="Thailand"]
```

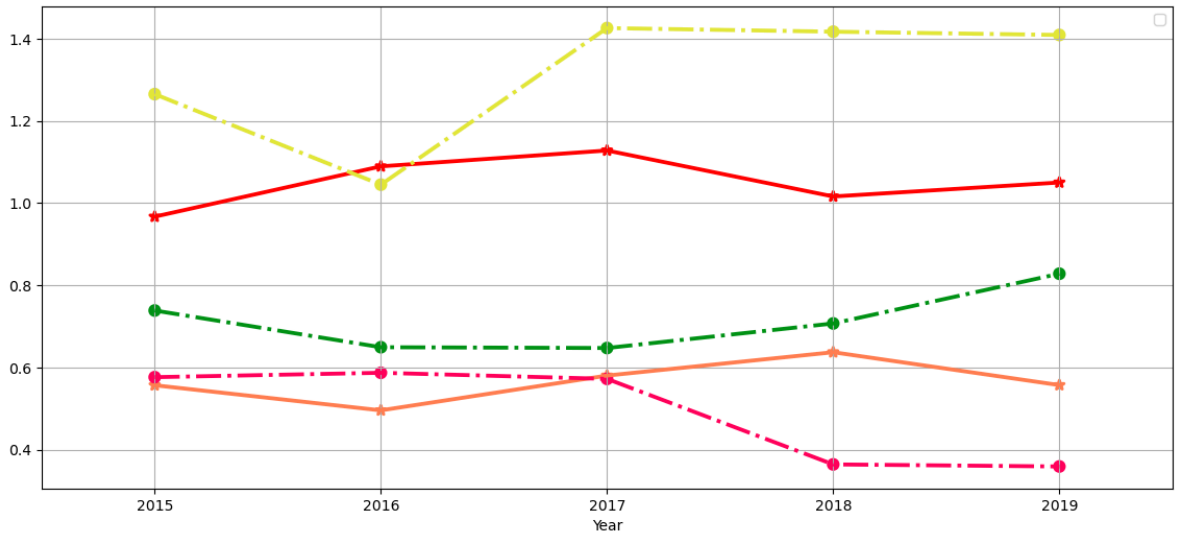
In [151]:
```
Thai_data = pd.concat([T_15,T_16,T_17,T_18,T_19])
Thai_data.head()
```

Out[151]:

	Country	Region	Happiness_Rank	Happiness_Score	Standard Error	GDP_per_Capita	Soc
33	Thailand	Southeastern Asia	34	6.455	0.03557	0.966900	1.265(
32	Thailand	Southeastern Asia	33	6.474	NaN	1.089300	1.0447
31	Thailand	NaN	32	6.424	NaN	1.127869	1.4257
45	Thailand	NaN	46	6.072	NaN	1.016000	1.417(
51	Thailand	NaN	52	6.008	NaN	1.050000	1.409(

In [152]:
```
plt.figure(figsize=(14,6))
sns.pointplot(x="Year", y="GDP_per_Capita", data=Thai_data, color="#FF0000",ma
sns.pointplot(x="Year", y="Healthy_life_expectancy", data=Thai_data, color="#0
sns.pointplot(x="Year", y="Freedom", data=Thai_data, color="#FF7F50",markers='
sns.pointplot(x="Year", y="Social", data=Thai_data, color="#E4E73A",linestyles
sns.pointplot(x="Year", y="Generosity", data=Thai_data, color="#FF005D",linest
plt.ylabel('')
plt.grid()
plt.legend()
plt.show()
```

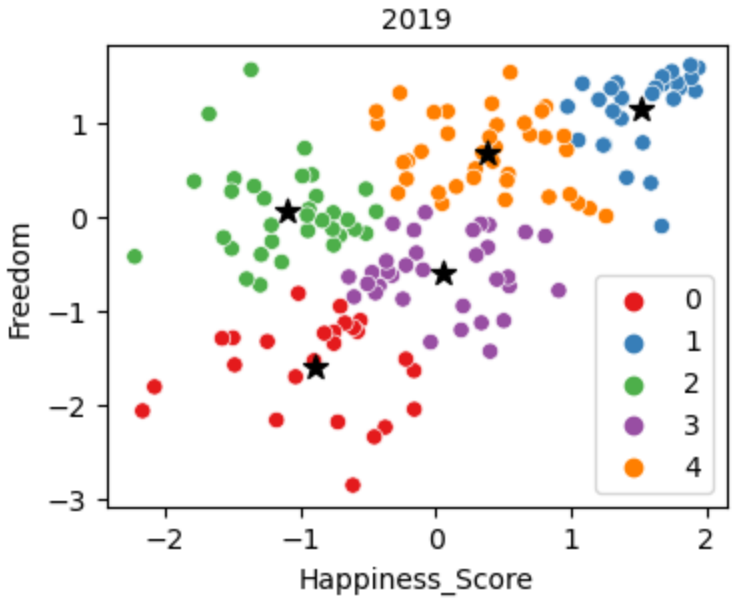
```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



In [153]:
```
df2= y_2015[['Happiness_Score','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['Happiness_Score','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='Happiness_Score', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

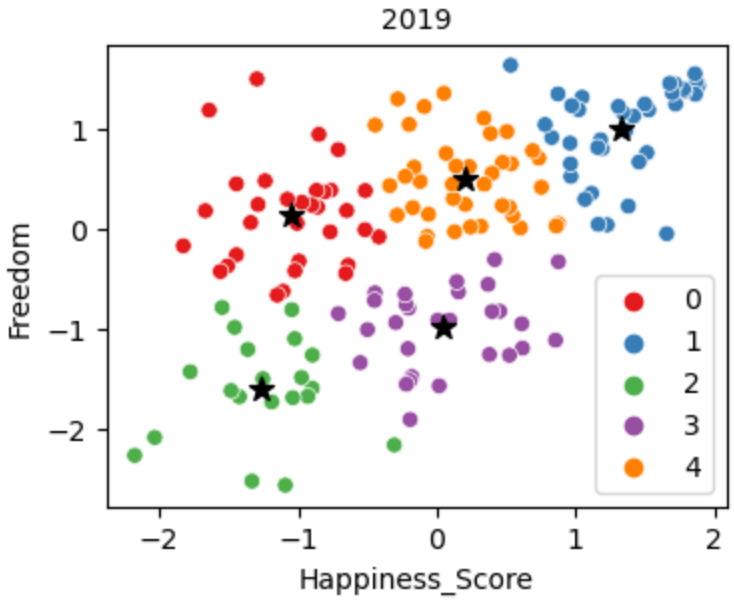
Out[153]: Text(0.5, 1.0, '2019')



In [154]:
```
df2= y_2016[['Happiness_Score','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['Happiness_Score','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='Happiness_Score', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

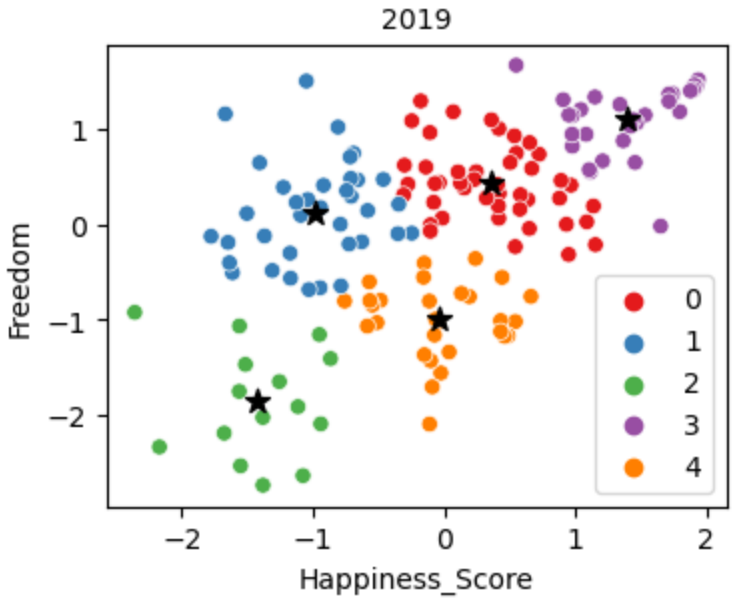
Out[154]: Text(0.5, 1.0, '2019')



In [106]:
```
df2= y_2017[['Happiness_Score','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['Happiness_Score','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='Happiness_Score', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

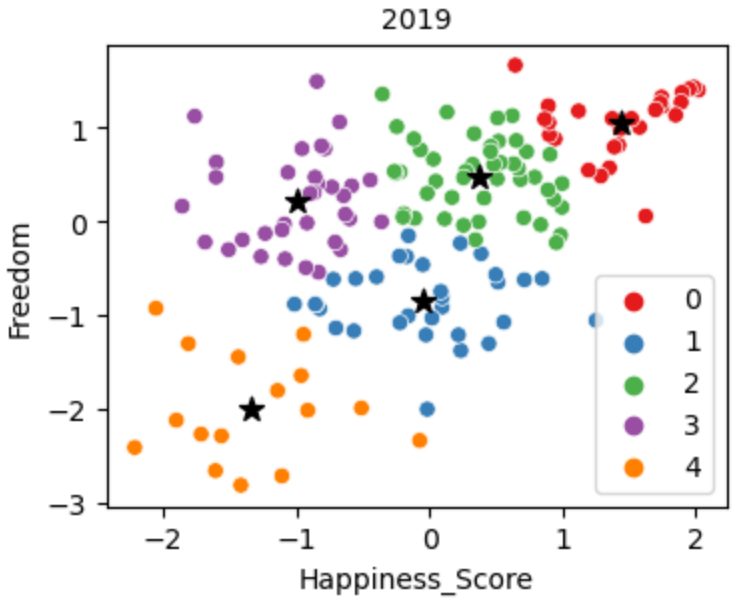
Out[106]: Text(0.5, 1.0, '2019')



In [155]:
```
df2= y_2018[['Happiness_Score','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['Happiness_Score','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='Happiness_Score', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

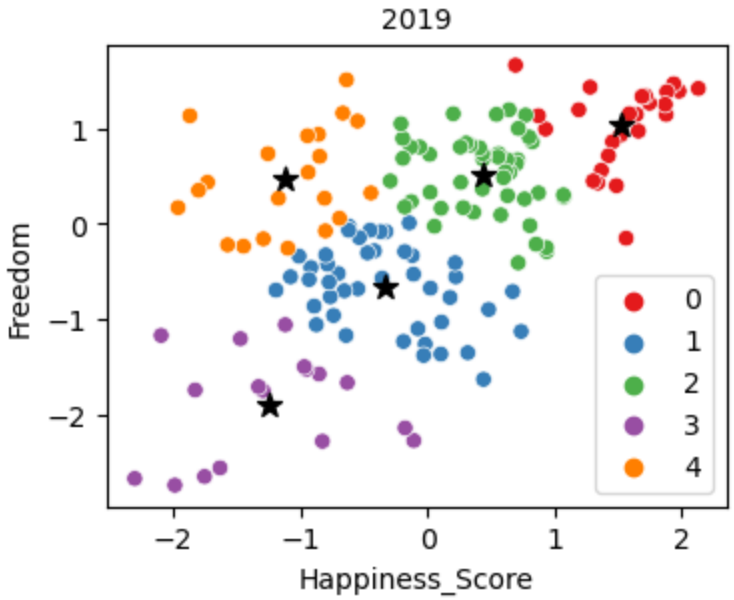
Out[155]: Text(0.5, 1.0, '2019')



In [156]:
```
df2= y_2019[['Happiness_Score','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['Happiness_Score','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='Happiness_Score', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

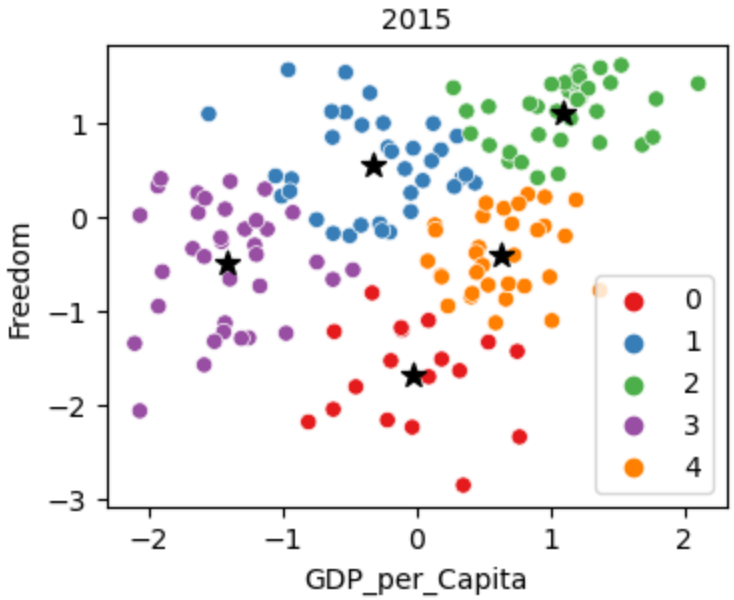
Out[156]: Text(0.5, 1.0, '2019')



In [157]:
```
df2= y_2015[['GDP_per_Capita','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['GDP_per_Capita','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='GDP_per_Capita', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2015',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

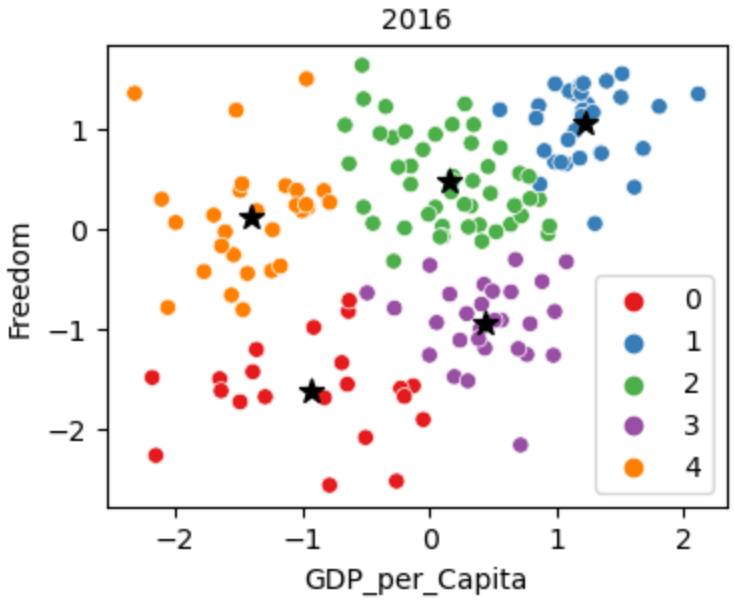
Out[157]: Text(0.5, 1.0, '2015')



In [112]:
```
df2= y_2016[['GDP_per_Capita','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['GDP_per_Capita','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='GDP_per_Capita', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2016',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

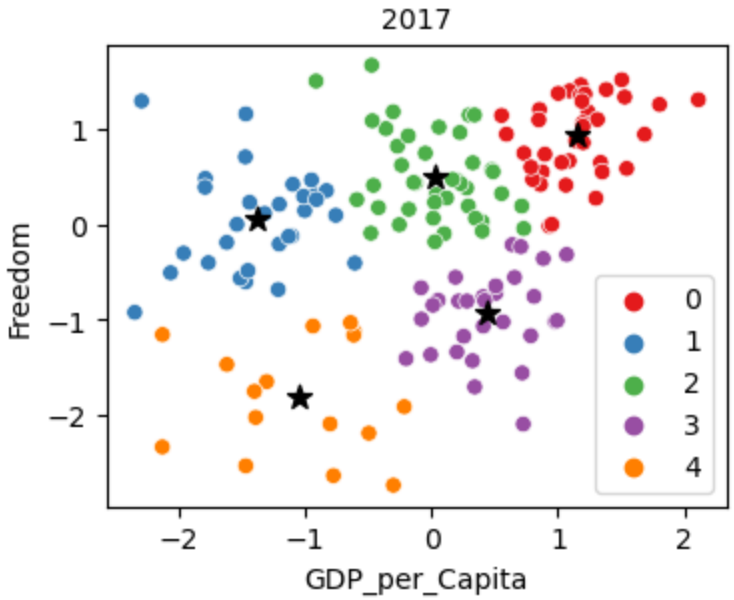
Out[112]: Text(0.5, 1.0, '2016')



In [158]:
```
df2= y_2017[['GDP_per_Capita','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['GDP_per_Capita','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='GDP_per_Capita', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2017',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

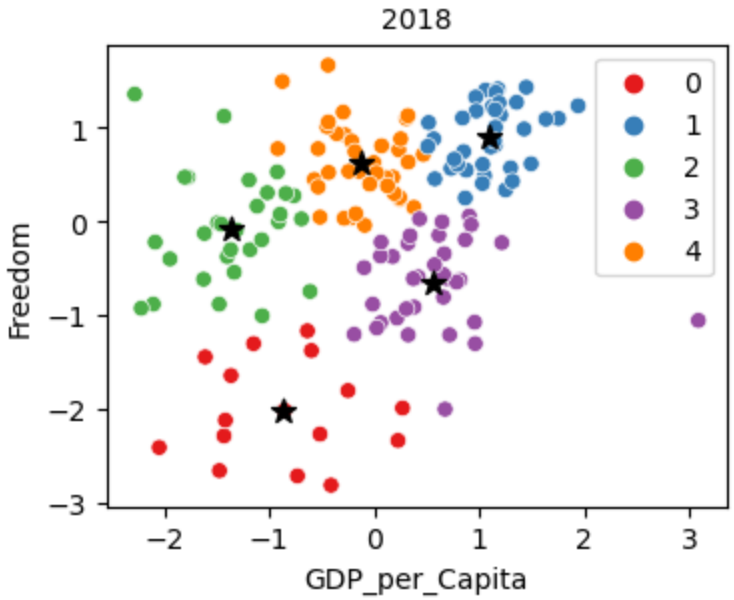
Out[158]: Text(0.5, 1.0, '2017')



In [114]:
```
df2= y_2018[['GDP_per_Capita','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['GDP_per_Capita','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='GDP_per_Capita', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2018',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[114]: Text(0.5, 1.0, '2018')



In [159]:
```
df2= y_2019[['GDP_per_Capita','Freedom']].dropna()

from sklearn.preprocessing import  StandardScaler
scaler = StandardScaler()

scaler.fit(df2)
df2[['GDP_per_Capita','Freedom']] = scaler.transform(df2)

from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5 ,random_state = 0)
model.fit(df2)
plt.figure(figsize=[4,3])
sns.scatterplot(data = df2,x='GDP_per_Capita', y='Freedom'
                        ,hue = model.labels_, palette ='Set1')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],
                color='k',marker ='*',s=80)
plt.title('2019',size=10)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[159]: Text(0.5, 1.0, '2019')