In [96]:
```
#Paul Galvez
#DSC 540
#Week 9/10
#Date: 5/12/23
```

In [97]:
```
#1. Data Wrangling with Python: Activity 9, page 294
```

In [98]:
```
#1 - import the necessary libraries including regex and beautifulsoup

import urllib.request, urllib.parse, urllib.error
import requests
from bs4 import BeautifulSoup
import ssl
import re
```

In [99]:
```
#2 - Check the SSL certificate - the SSL errors are to be ignored.
#the output is 2

check_ssl = ssl.create_default_context()
check_ssl.check_hostname = False
check_ssl.verify_mode - ssl.CERT_NONE
```

Out[99]:
```
2
```

In [100]:
```
#3 Read the HTML from the URL
#setting the HTML to variable url. The HTML and url is passed through beautifulsoup

url = 'https://www.gutenberg.org/browse/scores/top'
response = requests.get(url)
```

In [101]:
```
#4 - Write a small function to check the status of web request

def status_check(r):
    if r.status_code==200:
        print("Success!")
        return 1
    else:
        print("Failed!")
        return -1
```

In [102]:
```
#checking the response status of the web request - the request was a Success!

status_check(response)
```

```
Success!
```

Out[102]:
```
1
```

In [103]:
```
#5 - Decode the response and pass on to BeautifulSoup for HTML parsing
#setting url_content for parsing

url_content = response.content.decode(response.encoding)
```

In [104]:
```
#setting url_soup for beautifulsoup html parser

url_soup = BeautifulSoup(url_content, 'html.parser')
```

In [105]:
```
#6 - FInd all the href tags and store them in the list of links. Check what the list looks like -
#print the first 30 elements.
```

In [106]:
```
#creating empty list for HTML links

my_list = []
```

In [107]:
```
#creating a for loop to iterate over the my_list empty list for the HTML/URL

for link in url_soup.find_all('a'):
    my_list.append(link.get('href'))
```

In [108]:
```
#printing to show to the first 30 elements of the list for the href tags

my_list[:30]
```

Out[108]:
```
['/',
 '/about/',
 '/about/',
 '/policy/collection_development.html',
 '/about/contact_information.html',
 '/about/background/',
 '/policy/permission.html',
 '/policy/privacy_policy.html',
 '/policy/terms_of_use.html',
 '/ebooks/',
 '/ebooks/',
 '/ebooks/bookshelf/',
 '/browse/scores/top',
 '/ebooks/offline_catalogs.html',
 '/help/',
 '/help/',
 '/help/copyright.html',
 '/help/errata.html',
 '/help/file_formats.html',
 '/help/faq.html',
 '/policy/',
 '/help/public_domain_ebook_submission.html',
 '/help/submitting_your_own_work.html',
 '/help/mobile.html',
 '/attic/',
 '/donate/',
 '/donate/',
 '#books-last1',
 '#authors-last1',
 '#books-last7']
```

In [109]:
```
#7 - Use regular expression to find the numeric digits in these links.
#These are the file number for the Top 100 books.
```

In [110]:
```
#8 - setting empty list for the top 100 books - variable is top_100_books
#Initialize empty list to hold the file numbers

top_100_books = []
```

In [111]:
```
#Looping over the list. Number 19 to 118 are the links to the top 100 e books \
#the for loop will iterate over the range

for i in range(19,119):
    link=my_list[i]
    link=link.strip()
    n=re.findall('[0-9]+',link)
    if len(n)==1:
        top_100_books.append(int(n[0]))
```

In [112]:
```
#printing the list for the top 100 ebooks.

print ("\nThe file numbers for the top 100 ebooks are:\n"+"-"*70)
print(top_100_books)
```

```

The file numbers for the top 100 ebooks are:
----------------------------------------------------------------------
[1, 1, 7, 7, 30, 30, 1513, 2701, 2641, 145, 84, 37106, 100, 67979, 16389, 1342, 394, 6761, 6593, 4085, 2
160, 1259, 5197, 47629, 43, 20228, 64317, 11, 844, 174, 98, 2542, 345, 70769, 1400, 46, 5200, 70768, 166
1, 1080, 10676, 42108, 25344, 1184, 5000, 70766, 1260, 28054, 55, 1952, 6130, 15845, 14328, 120, 4300, 1
727, 2591, 76, 35899, 2554, 996, 2600, 74, 1232, 768, 27827, 45, 58585, 23042, 36, 30254, 205, 2680, 40
8, 2852, 4363, 16, 1399, 2500, 1998, 1533, 132, 5740, 244, 161, 236, 2814, 514, 600, 1497, 67098, 219]
```

In [113]:
```
#9 - What does the soup objects text look like? Use the .text method and print only the first
#2000 characters (do not print the whole thing as it is too long.)
```

In [114]:
```
#printing the first 2000 characters from url_soup

print(url_soup.text[:2000])
```

```
Collection Development
Contact Us
History & Philosophy
Permissions & License
Privacy Policy
Terms of Use



Search and Browse
          ▾


▾



Book Search
Bookshelves
Frequently Downloaded
Offline Catalogs
```

In [115]:
```
#10 - Search in the extracted text (using a reg. expression) from the soup object to find the
#names of the top 100 eBooks (yesterdays ranking.)
```

In [116]:
```
#creating empty list for eBooks

title_list = []
```

In [117]:
```
#11 - Create a starting index. It should point at the text "Top 100 Ebooks yesterday".
#Hint: Use splitlines() method of the soup.text.
#It splits the lines of the text of the soup object.
#setting my_index for empty list (title_list)

my_indx=url_soup.text.splitlines().index('Top 100 EBooks yesterday')
```

In [118]:
```
#Loop 1-100 to add the strings of next 100 lines to this temporary list.
#Hint: splitlines() method
#Looping through to the empty list.

for i in range(100):
    title_list.append(url_soup.text.splitlines()[my_indx+2+i])
```

In [119]:
```
#12 - Use regular expression to extract only text from the name strings and append to an empty list
#Hint: Use match and span to find indices and use them
#trying to loop...getting an error message and not sure why?

my_list=[]
for i in range(100):
    id1,id2=re.match('^[a-zA-Z ]*',my_list[i]).span()
    lst_titles.append(my_list[i][id1:id2])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_10404\4133264883.py in <module>
      5 my_list=[]
      6 for i in range(100):
----> 7     id1,id2=re.match('^[a-zA-Z ]*',my_list[i]).span()
      8     lst_titles.append(my_list[i][id1:id2])

IndexError: list index out of range
```

In [ ]:

In [120]:
```
#2. Data Wrangling with Python: Activity 10, page 295
```

In [121]:
```
#1 - import urllib.request, urllib.parse, urllib.error and json

import urllib.request, urllib.parse, urllib.error
import json
```

In [122]:
```
#2- Load the secret API key (you have to get one from OMDB website and use that,
#1000 daily limit) from a JSON file, stored in the same folder into a variable
#Hint: Use json.loads()

with open('APIkeys.json') as f:
    keys = json.load(f)
    omdbapi = keys['OMDBapi']
```

In [123]:
```
serviceurl = 'http://www.omdbapi.com/?'
apikey = '&apikey='+omdbapi
```

In [124]:
```
def print_json(json_data):
    list_keys=['Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre', 'Director', 'Writer',
               'Actors', 'Plot', 'Language', 'Country', 'Awards', 'Ratings',
               'Metascore', 'imdbRating', 'imdbVotes', 'imdbID','Poster']
    print("-"*50)
    for k in list_keys:
        if k in list(json_data.keys()):
            print(f"{k}: {json_data[k]}")
    print("-"*50)
```

In [146]:
```
def search_movie(title):
    try:
        url = serviceurl + urllib.parse.urlencode({'t': str(title)})+apikey
        print(f'Retrieving the data of "{title}" now... ')
        print(url)
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)

        if json_data['Response']=='True':
            print_json(json_data)
            if json_data['Poster']!='N/A':
                save_poster(json_data)
        else:
            print("Error encountered: ",json_data['Error'])

    except urllib.error.URLError as e:
        print(f"ERROR: {e.reason}")
```

In [164]:
```
search_movie("Captain Marvel")
```

```
Retrieving the data of "Captain Marvel" now...
http://www.omdbapi.com/?t=Captain+Marvel&apikey=4c776206 (http://www.omdbapi.com/?t=Captain+Marvel&apike
y=4c776206)
--------------------------------------------------
Title: Captain Marvel
Year: 2019
Rated: PG-13
Released: 08 Mar 2019
Runtime: 123 min
Genre: Action, Adventure, Sci-Fi
Director: Anna Boden, Ryan Fleck
Writer: Anna Boden, Ryan Fleck, Geneva Robertson-Dworet
Actors: Brie Larson, Samuel L. Jackson, Ben Mendelsohn
Plot: Carol Danvers becomes one of the universe's most powerful heroes when Earth is caught in the middl
e of a galactic war between two alien races.
Language: English
Country: United States, Australia
Awards: 9 wins & 56 nominations
Ratings: [{'Source': 'Internet Movie Database', 'Value': '6.8/10'}, {'Source': 'Rotten Tomatoes', 'Valu
e': '79%'}, {'Source': 'Metacritic', 'Value': '64/100'}]
Metascore: 64
imdbRating: 6.8
imdbVotes: 576,344
imdbID: tt4154664
Poster: https://m.media-amazon.com/images/M/MV5BMTE0YWFmOTMtYTU2ZS00ZTIxLWE3OTEtYTNiYzBkZjViZThiXkEyXkFq
cGdeQXVyODMzMzQ4OTI@._V1_SX300.jpg (https://m.media-amazon.com/images/M/MV5BMTE0YWFmOTMtYTU2ZS00ZTIxLWE3
OTEtYTNiYzBkZjViZThiXkEyXkFqcGdeQXVyODMzMzQ4OTI@._V1_SX300.jpg)
--------------------------------------------------
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_10404\4193267268.py in <module>
----> 1 search_movie("Captain Marvel")

~\AppData\Local\Temp\ipykernel_10404\2257273870.py in search_movie(title)
     11             print_json(json_data)
     12             if json_data['Poster']!='N/A':
---> 13                 save_poster(json_data)
     14         else:
     15             print("Error encountered: ",json_data['Error'])

NameError: name 'save_poster' is not defined
```

In [127]:

```
search_movie("Random_error")
```

```
Retrieving the data of "Random_error" now...
http://www.omdbapi.com/?t=Random_error&apikey=4c776206 (http://www.omdbapi.com/?t=Random_error&apikey=4c
776206)
Error encountered:  Movie not found!
```

In [ ]:

In [128]:

```
#3. Connect to an API of your choice and do a simple data pull - you can use any API -
#except the API you have selected for your project.

from urllib.error import HTTPError, URLError
import pandas as pd
```

In [158]:
```
#I pulled data from the FBI wanted list and printed the results below. My parameters were the set
#page and number. I connected to a free API from https://www.fbi.gov/wanted/api from the most
#wanted list. Changing the page number will pull the correlated data to that person on the FBI list.

response = requests.get('https://api.fbi.gov/wanted/v1/list', params={
    'page': 10
})
data = json.loads(response.content)
print(data['page'])
print(data['items'][0]['title'])
```

```
10
MIGUEL ANGEL MORALES OROZCO
```

In [ ]:
```

```

In [130]:
```
#4. Using one of the datasets provided in Weeks 7 & 8, or a dataset of your own,
#choose 3 of the following visualizations to complete.
#You must submit via PDF along with your code.
#You are free to use Matplotlib, Seaborn or another package if you prefer.
#a. Line
#b. Scatter
#c. Bar
#d. Histogram
#e. Density Plot
#f. Pie Chart
```

In [131]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import pandas as pd
from cycler import cycler
import seaborn as sns
```

In [132]:
```
df = pd.read_csv('2019.csv')
df
```

Out[132]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

156 rows × 9 columns

In [133]:
```
df.columns
```

Out[133]:
```
Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

In [134]:
```
df.head()
```

Out[134]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [135]:
```
df.tail()
```

Out[135]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

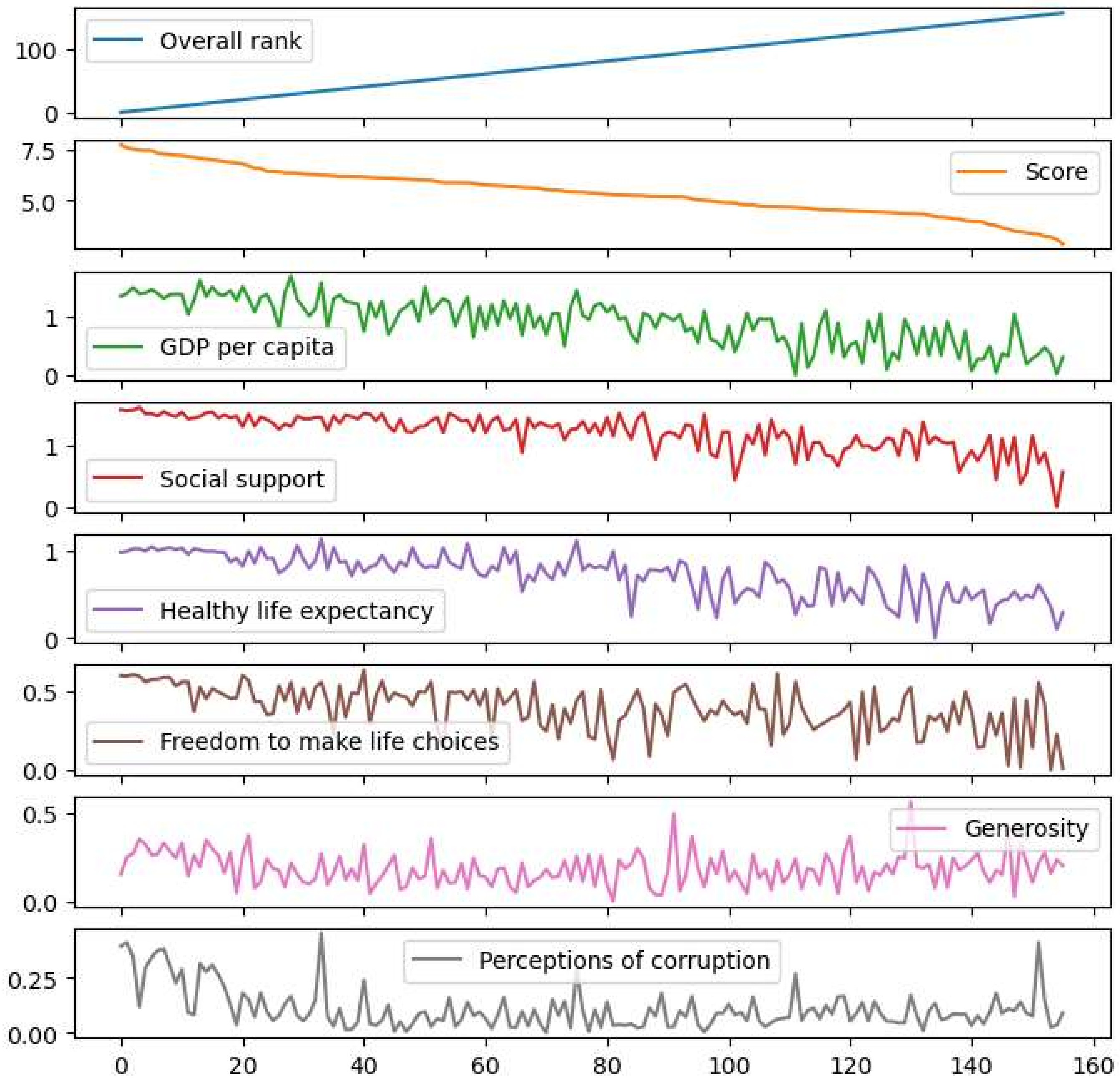
In [136]:
```
df.describe()
```

Out[136]:

	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
count	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000
mean	78.500000	5.407096	0.905147	1.208814	0.725244	0.392571	0.184846	0.110603
std	45.177428	1.113120	0.398389	0.299191	0.242124	0.143289	0.095254	0.094538
min	1.000000	2.853000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.750000	4.544500	0.602750	1.055750	0.547750	0.308000	0.108750	0.047000
50%	78.500000	5.379500	0.960000	1.271500	0.789000	0.417000	0.177500	0.085500
75%	117.250000	6.184500	1.232500	1.452500	0.881750	0.507250	0.248250	0.141250
max	156.000000	7.769000	1.684000	1.624000	1.141000	0.631000	0.566000	0.453000

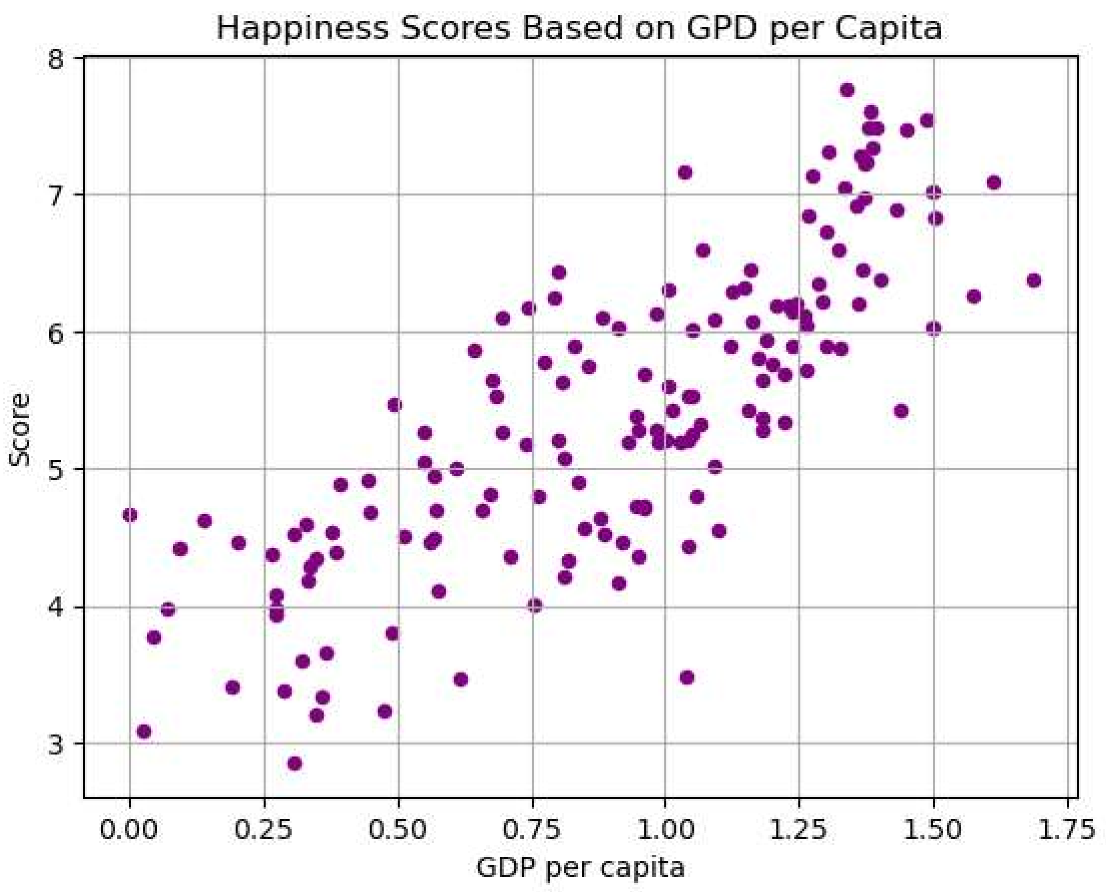
In [137]:

```
#Line plots showing the rate of happiness versus the variables or columns from the dataset.

df.plot(subplots=True, figsize=(8, 8));
```

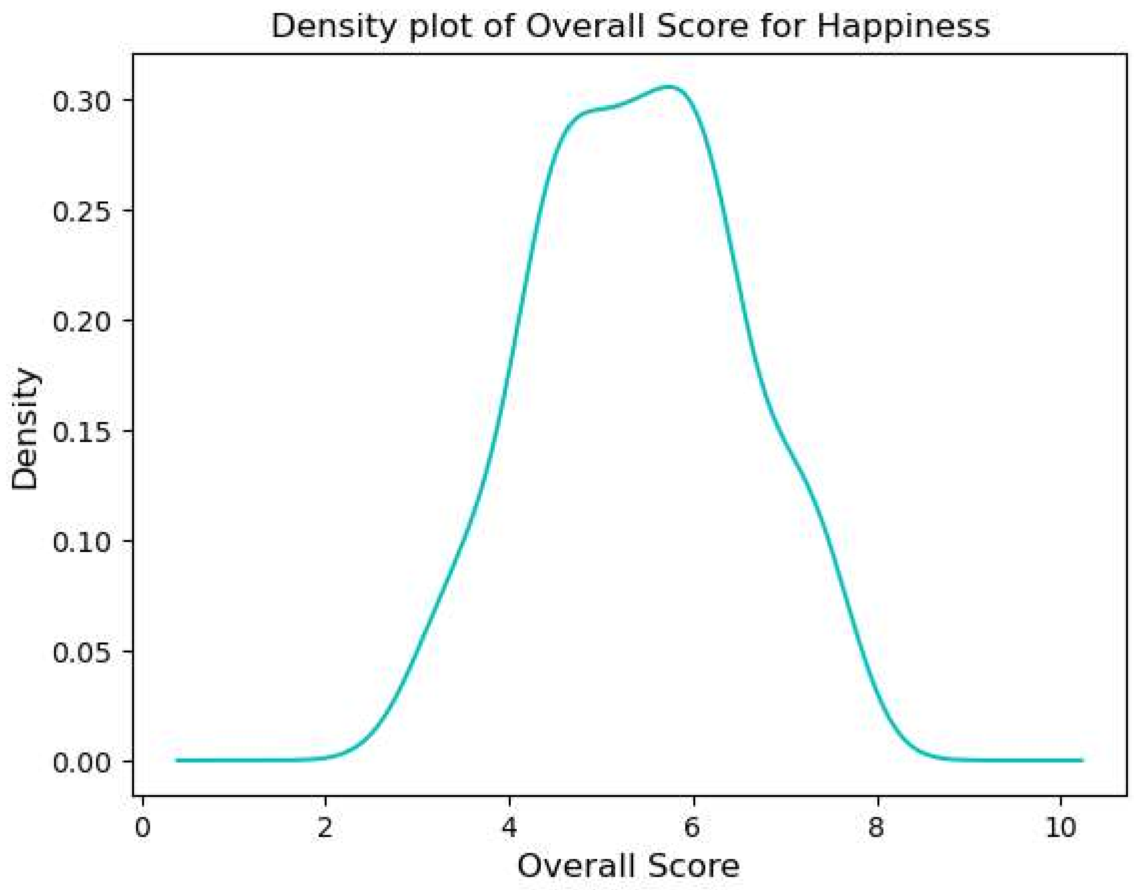

Overall rank
Score
GDP per capita
Social support
Healthy life expectancy
Freedom to make life choices
Generosity
Perceptions of corruption
100
0
7.5
5.0
1
0
1
0
1
0
0.5
0.0
0.5
0.0
0.25
0.00
0
20
40
60
80
100
120
140
160

In [138]:
```
#scatter plot for happiness index and scores based on GDP. We can see that as GPD rises, scores for overa
#go up as well.

df.plot.scatter(x='GDP per capita', y='Score', color = 'purple');
plt.grid()
plt.title('Happiness Scores Based on GPD per Capita')
```

Out[138]:
```
Text(0.5, 1.0, 'Happiness Scores Based on GPD per Capita')
```



In [139]:
```
#density plot of overall happiness score

df.Score.plot.density(color='c')
plt.title('Density plot of Overall Score for Happiness', fontsize=12)
plt.xlabel('Overall Score',fontsize = 12)
plt.ylabel('Density',fontsize = 12)
plt.show()
```



In [140]:
```
happy_scores = {'Finland': 7.9, 'Denmark': 7.6, 'Norway': 7.5, 'Iceland': 7.3, 'Netherlands': 7.4, 'Rwand
       'Afghanistan': 3.2, 'Central African Republic': 3.0, 'South Sudan': 2.8}
happy_scores
```

Out[140]:
```
{'Finland': 7.9,
 'Denmark': 7.6,
 'Norway': 7.5,
 'Iceland': 7.3,
 'Netherlands': 7.4,
 'Rwanda': 3.3,
 'Tanzania': 3.2,
 'Afghanistan': 3.2,
 'Central African Republic': 3.0,
 'South Sudan': 2.8}
```

In [141]:
```
new_df1 = df[["Score", "GDP per capita"]]
new_df1
```

Out[141]:

	Score	GDP per capita
0	7.769	1.340
1	7.600	1.383
2	7.554	1.488
3	7.494	1.380
4	7.488	1.396
...	...	...
151	3.334	0.359
152	3.231	0.476
153	3.203	0.350
154	3.083	0.026
155	2.853	0.306

156 rows × 2 columns

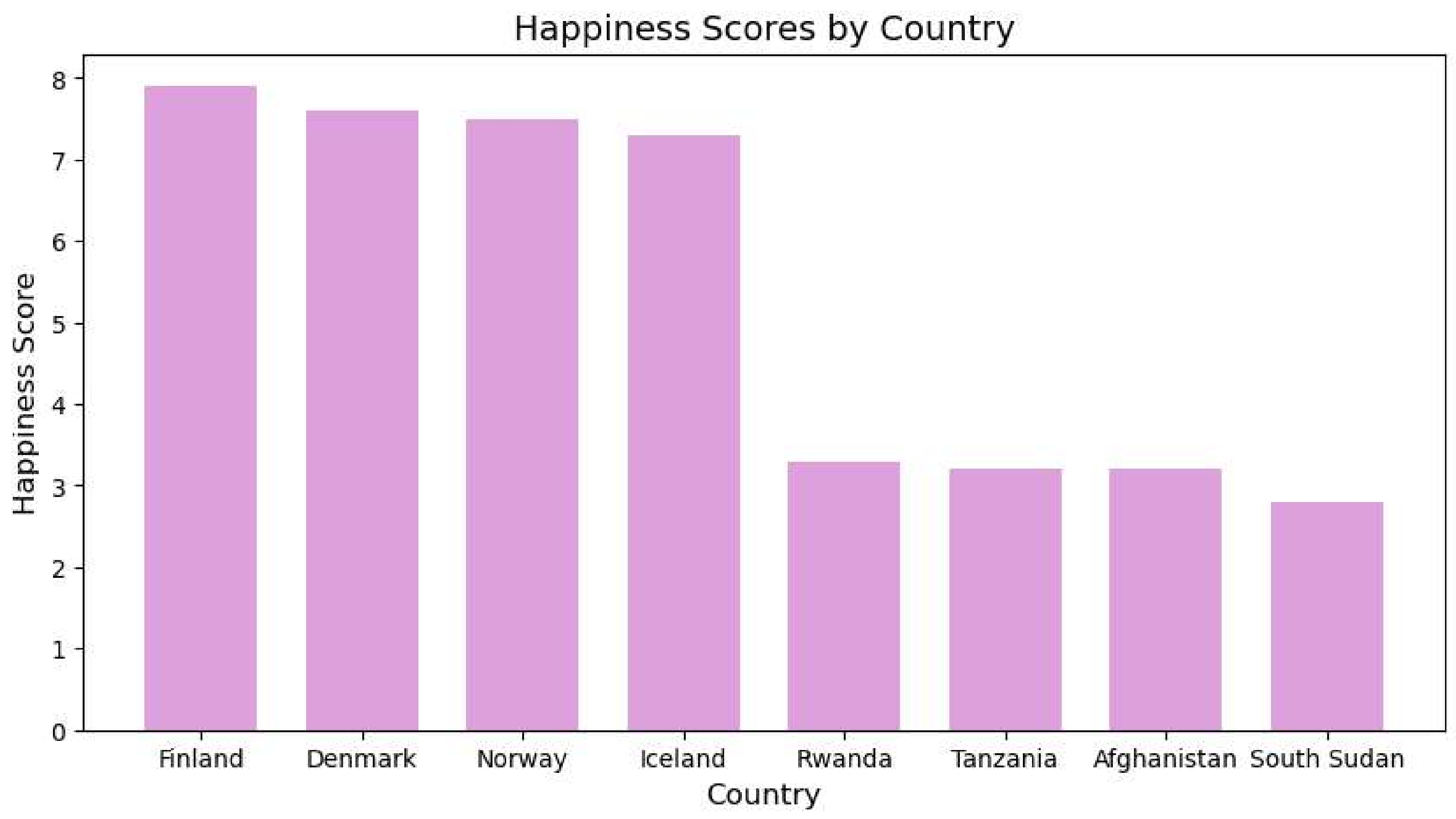
In [145]:
```
other_df3 = df[['Country or region', 'Score' ]]
other_df3
```

Out[145]:

	Country or region	Score
0	Finland	7.769
1	Denmark	7.600
2	Norway	7.554
3	Iceland	7.494
4	Netherlands	7.488
...	...	...
151	Rwanda	3.334
152	Tanzania	3.231
153	Afghanistan	3.203
154	Central African Republic	3.083
155	South Sudan	2.853

In [143]:

```
#bar chart for happiness scores by country. Here are the tops and bottoms of the dataset. The score is on
#1 being the worst possible and 10 being the highest possible score.

data = {'Finland': 7.9, 'Denmark': 7.6, 'Norway': 7.5, 'Iceland': 7.3, 'Rwanda': 3.3, 'Tanzania': 3.2,
       'Afghanistan': 3.2, 'South Sudan': 2.8}
Country = list(data.keys())
Score = list(data.values())

fig = plt.figure(figsize = (10, 5))

plt.bar(Country, Score, color ='plum',
        width = 0.7)

plt.xlabel("Country", fontsize=12)
plt.ylabel("Happiness Score", fontsize=12)
plt.title("Happiness Scores by Country", fontsize=14)
plt.show()
```


Happiness Scores by Country
Happiness Score
0
1
2
3
4
5
6
7
8
Finland
Denmark
Norway
Iceland
Rwanda
Tanzania
Afghanistan
South Sudan
Country

```
In [ ]:
```