Combining Datasets and Saving them to a Database File

KJ MoChroi
DSC 540 Fall 2022
Bellevue University

Change Control Log:

Change#: 1
Change(s) Made: Imported data from three sources and cleaned it.
Date of Change: 11/16/2022
Author: KJ MoChroi Change Approved by: KJ MoChroi
Date Moved to Production: 11/19/2022

Change#: 2
Change(s) Made: Converted dataframes to tables in SQL database and merged those tables.
Date of Change: 11/17/2022
Author: KJ MoChroi
Change Approved by: KJ MoChroi
Date Moved to Production: 11/19/2022

Change#: 3
Change(s) Made: produced visualizations.
Date of Change: 11/17/2022
Author: KJ MoChroi
Change Approved by: KJ MoChroi
Date Moved to Production: 11/19/2022

Change#: 4
Change(s) Made: Completed Write-up.
Date of Change: 11/17/2022
Author: KJ MoChroi
Change Approved by: KJ MoChroi
Date Moved to Production: 11/19/2022

API Data

In [1]:
```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
```

In [2]:
```
# See if we can connect to API
response = requests.get("https://world-happiness-database.herokuapp.com/api/happiness")
print(response.status_code)
```

```
200
```

In [3]:
```
# Extract Data from the Request using the Json() Method
json = response.json()
```

In [4]:
```
# Turn json into dataframe
api_df = pd.DataFrame(json)
```

In [5]:
```
# Drop columns
api_df = api_df[['country', 'happiness']]
```

In [6]:
```
# Rename columns
api_df.columns = ['Country', 'Happiness']
```

In [7]:
```
# sort alphabetically by Country
api_df.sort_values('Country', inplace=True)
```

In [8]:
```
# Alter names of countries so tables can join
api_df["Country"]= api_df["Country"].replace("Czech Republic","Czechia")
```

In [9]:
```
api_df["Country"]= api_df["Country"].replace("Palestinian Territories","Palestine")
```

HTML Data

In [10]:
```
# Libraries
from bs4 import BeautifulSoup
```

In [11]:
```
# Create an URL object
url = 'https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)'
```

In [12]:
```
# Create object page
page = requests.get(url)
```

In [13]:
```
# parser-lxml = Change html to Python friendly format
# Obtain page's information
soup = BeautifulSoup(page.text, 'lxml')
```

In [14]:
```
table = soup.find("table", {"class": "wikitable"})
```

In [15]:
```
# Obtain column names with tag <th>
headers = []
for i in table.find_all('th'):
    title = i.text
    headers.append(title)
```

In [16]:
```
headers = ['Country',
 'UN Region',
 'IMF Estimate',
'IMF Year',
 'World Bank Estimate',
'World Bank Year',
 'United Nations Estimate',
'United Nations Year'
]
# The previous headers didn't make sense because of the way the cells are merged on the
```

In [17]:
```
# Turn table into a dataframe for easier reading.
html_df = pd.DataFrame(columns = headers)
```

In [18]:
```
# Create a for loop to fill dataframe
for j in table.find_all('tr')[3:]:
    row_data= j.find_all('td')
    row = [i.text for i in row_data]
    holder = []
    for item in row:
        holder.append(item)
        if item == '—' or item == '—\n':
            holder.append('—')  # This is for rows with merged cells on webpage
    length = len(html_df)
    html_df.loc[length] = holder
```

In [19]:
```
html_df.drop(['IMF Year', 'World Bank Year', 'United Nations Year'], axis=1, inplace=Tr
```

In [20]:
```
# Remove sepcial characters
html_df['IMF Estimate'] = html_df['IMF Estimate'].str.replace('\W', '', regex=True)
html_df['World Bank Estimate'] = html_df['World Bank Estimate'].str.replace('\W', '', r
html_df['United Nations Estimate'] = html_df['United Nations Estimate'].str.replace('\W
```

In [21]:
```
# convert estimates to numbers
html_df['IMF Estimate'] = pd.to_numeric(html_df['IMF Estimate'], errors='coerce')
html_df['World Bank Estimate'] = pd.to_numeric(html_df['World Bank Estimate'], errors='
html_df['United Nations Estimate'] = pd.to_numeric(html_df['United Nations Estimate'],
```

In [22]:
```
# create new data column
html_df['Mean_GDP'] = html_df[['IMF Estimate', 'World Bank Estimate', 'United Nations E
```

In [23]:
```
# drop columns
html_df = html_df[['Country', 'Mean_GDP']]
```

In [24]:
```
# sort alphabetically by Country
html_df.sort_values('Country', inplace=True)
```

In [25]:
```
html_df["Country"]= html_df["Country"].replace("\xa0","")
# Even after doing this step, these characters still show up in the SQL database after
```

Flat File Data

In [26]:
```
GDI_df = pd.read_csv('GDI.csv')
```

In [27]:
```
# Drop unnecesary columns
GDI_df = GDI_df[['Country', 'GDI_Value', 'Lif_Expec_Female', 'Lif_Excep_Male']]
```

In [28]:
```
# Convert columns to numerical and replace strings with NaN
GDI_df['GDI_Value'] = GDI_df['GDI_Value'].apply(lambda x: pd.to_numeric(x, errors='coer
GDI_df['GDI_Value'].isna().sum()
```

Out[28]:
```
22
```

In [29]:
```
GDI_df['Lif_Expec_Female'] = GDI_df['Lif_Expec_Female'].apply(lambda x: pd.to_numeric(x
GDI_df['Lif_Expec_Female'].isna().sum()
```

Out[29]:
```
6
```

In [30]:
```
GDI_df['Lif_Excep_Male'] = GDI_df['Lif_Excep_Male'].apply(lambda x: pd.to_numeric(x, er
GDI_df['Lif_Excep_Male'].isna().sum()
```

Out[30]:
```
6
```

In [31]:
```
# Remove new NaN values from tables
GDI_df = GDI_df.dropna()
```

In [32]:
```
# sort alphabetically by Country
GDI_df.sort_values('Country', inplace=True)
```

In [33]:
```
# Alter names of countries so tables can join
GDI_df["Country"]= GDI_df["Country"].replace("Bolivia (Plurinational State of)", "Boliv
```

In [34]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Iran (Islamic Republic of)","Iran")
```

In [35]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Lao People's Democratic Republic","Laos")
```

In [36]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Moldova (Republic of)","Moldova")
```

In [37]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Palestine, State of","Palestine")
```

In [38]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Russian Federation","Russia")
```

In [39]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Korea (Republic of)","South Korea")
```

In [40]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Syrian Arab Republic","Syria")
```

In [41]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Tanzania (United Republic of)","Tanzania"
```

In [42]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Venezuela (Bolivarian Republic of)","Vene
```

In [43]:
```
GDI_df["Country"]= GDI_df["Country"].replace("Viet Nam","Vietnam")
```

SQL Database

In [44]:
```
import sqlite3
```

In [45]:
```
#make connection to database
conn = sqlite3.connect('final_project.db')
```

In [46]:
```
# Convert dataframes into sql table
GDI_df.to_sql('gender', conn, if_exists='replace')
```

In [47]:
```
api_df.to_sql('happiness', conn, if_exists='replace')
```

In [48]:
```
html_df.to_sql('GDP', conn, if_exists='replace')
```

In [49]:
```
# Create cursor object
cursor = conn.cursor()
```

In [50]:
```
# selecting table for removing extra characters from strings:
sqlselect = '''
SELECT Country, Mean_GDP
FROM GDP
'''
```

In [51]:
```
# Executing the query
cursor.execute(sqlselect)
```

Out[51]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [52]:
```
# removing extra characters from strings that appeared when converting from df to table
sqlreplace = '''
UPDATE GDP
SET Country = REPLACE(Country, '\xa0', '')
'''
```

In [53]:
```
# Executing the query
cursor.execute(sqlreplace)
```

Out[53]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [54]:
```
# to avoid error:
sql2 = '''DROP TABLE IF EXISTS joined_table;'''
```

In [55]:
```
# Executing the query
cursor.execute(sql2)
```

Out[55]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [56]:
```
# Left Join both tables to gender table
sql3 = '''CREATE TABLE joined_table AS
SELECT gender.Country, gender.GDI_Value, gender.Lif_Expec_Female, gender.Lif_Excep_Male
FROM gender
LEFT JOIN happiness ON gender.Country = happiness.Country
LEFT JOIN GDP ON gender.Country = GDP.Country;
'''
```

In [57]:
```
# Executing the query
cursor.execute(sql3)
```

Out[57]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [58]:
```
# Dele rows with missing Happiness values
sqldelete = '''DELETE FROM joined_table
WHERE Happiness IS NULL;'''
```

In [59]:
```
# Executing the query
cursor.execute(sqldelete)
```

Out[59]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [60]:
```
# Dele rows with missing GDP values
sqldelete2 = '''DELETE FROM joined_table
WHERE Mean_GDP IS NULL;'''
```

In [61]:
```
# Executing the query
cursor.execute(sqldelete2)
```

Out[61]:
```
<sqlite3.Cursor at 0x1921547a2d0>
```

In [62]:
```
# select table
tableview = pd.read_sql_query("SELECT * FROM joined_table", conn)
```

In [63]:
```
# export to pandas dataframe for visualizations
df = pd.DataFrame(tableview, columns = ['Country', 'GDI_Value', 'Lif_Expec_Female', 'Li
```

In [72]:
```
# Closing the connection
conn.close()
```

Visualizations

In [65]:
```
# Human Readable Dataset
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',500)
pd.set_option('display.width',1000)
print(df)
```

```
                   Country  GDI_Value  Lif_Expec_Female  Lif_Excep_Male  Happiness
Mean_GDP
0              Afghanistan      0.659              66.4            63.4      2.694
2.001500e+04
1                  Albania      0.967              80.2            77.0      5.004
1.714200e+04
2                  Algeria      0.858              78.1            75.7      5.043
1.676090e+05
3                   Angola      0.903              64.0            58.4      3.795
8.654933e+04
4                Argentina      0.993              80.0            73.2      5.793
5.017527e+05
5                  Armenia      0.982              78.5            71.3      5.062
1.473167e+04
6                Australia      0.976              85.4            81.5      7.177
1.563640e+06
7                  Austria      0.964              83.9            79.2      7.396
4.594623e+05
```

8	Azerbaijan	0.943	75.5	70.5	5.168	5.576500e+04
9	Bahrain	0.922	78.4	76.4	6.227	3.877233e+04
10	Bangladesh	0.904	74.6	70.9	4.499	4.021667e+05
11	Belarus	1.007	79.6	69.7	5.234	6.939400e+04
12	Belgium	0.974	83.9	79.3	6.892	5.704103e+05
13	Belize	0.976	77.8	71.7	5.956	2.017333e+03
14	Benin	0.855	63.3	60.2	5.820	1.684600e+04
15	Bhutan	0.921	72.2	71.4	5.082	2.501667e+03
16	Bolivia	0.945	74.5	68.7	5.916	4.013733e+04
17	Bosnia and Herzegovina	0.937	79.9	74.9	5.887	2.201800e+04
18	Botswana	0.998	72.4	66.5	3.461	1.713367e+04
19	Brazil	0.993	79.6	72.2	6.191	1.649474e+06
20	Bulgaria	0.995	78.7	71.6	5.099	7.838900e+04
21	Burkina Faso	0.867	62.3	60.7	4.927	1.845833e+04
22	Burundi	0.999	63.4	59.8	3.775	3.329000e+03
23	Cambodia	0.922	71.9	67.5	5.122	2.686067e+04
24	Cameroon	0.864	60.6	58.0	5.251	4.311067e+04
25	Canada	0.986	84.4	80.4	7.175	1.945050e+06
26	Central African Republic	0.801	55.5	51.1	3.476	2.442667e+03
27	Chad	0.764	55.7	52.8	4.486	1.195967e+04
28	Chile	0.963	82.4	77.8	6.436	2.936217e+05
29	China	0.957	79.2	74.8	5.131	1.692602e+07
30	Colombia	0.989	80.0	74.5	5.984	3.095297e+05
31	Comoros	0.891	66.1	62.6	3.973	1.268333e+03
32	Costa Rica	0.981	82.9	77.7	7.141	6.476400e+04
33	Croatia	0.990	81.6	75.3	5.536	6.480733e+04
34	Cuba	0.944	80.8	76.8	5.418	1.073520e+05
35	Cyprus	0.979	83.0	78.9	6.276	2.634533e+04
36	Denmark	0.983	82.9	78.9	7.649	3.799710e+05
37	Dominican Republic	0.999	77.4	71.0	5.433	9.516867e+04

38	Ecuador	0.967	79.8	74.3	6.128	1.068143e+05
39	Egypt	0.882	74.4	69.7	4.005	4.141820e+05
40	El Salvador	0.975	77.8	68.5	6.276	2.845500e+04
41	Estonia	1.017	82.7	74.4	6.091	3.532233e+04
42	Ethiopia	0.837	68.5	64.7	4.379	1.063543e+05
43	Finland	0.990	84.7	79.1	7.858	2.834390e+05
44	France	0.987	85.5	79.7	6.666	2.781960e+06
45	Gabon	0.916	68.7	64.4	4.783	1.853367e+04
46	Gambia	0.846	63.5	60.7	4.922	2.026667e+03
47	Georgia	0.980	78.1	69.3	4.659	1.992133e+04
48	Germany	0.972	83.7	78.9	7.118	4.033560e+06
49	Ghana	0.911	65.2	63.0	5.481	7.404067e+04
50	Greece	0.963	84.7	79.8	5.409	2.090280e+05
51	Guatemala	0.941	77.2	71.4	6.627	8.496967e+04
52	Guinea	0.817	62.1	60.9	5.252	1.702867e+04
53	Guyana	0.961	73.1	66.9	5.993	9.239667e+03
54	Haiti	0.875	66.2	61.8	3.615	1.887700e+04
55	Honduras	0.978	77.6	73.0	5.908	2.762833e+04
56	Hungary	0.981	80.3	73.2	6.065	1.742467e+05
57	Iceland	0.969	84.5	81.5	7.476	2.495967e+04
58	India	0.820	71.0	68.5	3.818	3.102238e+06
59	Indonesia	0.940	74.0	69.6	5.340	1.177982e+06
60	Iran	0.866	77.9	75.6	4.278	1.048201e+06
61	Iraq	0.774	72.7	68.6	4.462	2.191740e+05
62	Ireland	0.981	83.9	80.7	6.962	4.814083e+05
63	Israel	0.973	84.5	81.3	6.927	4.719570e+05
64	Italy	0.968	85.5	81.3	6.517	1.995174e+06
65	Jamaica	0.994	76.1	72.9	5.890	1.451800e+04
66	Japan	0.978	87.7	81.5	5.794	4.765267e+06
67	Jordan	0.875	76.3	72.8	4.639	4.566900e+04

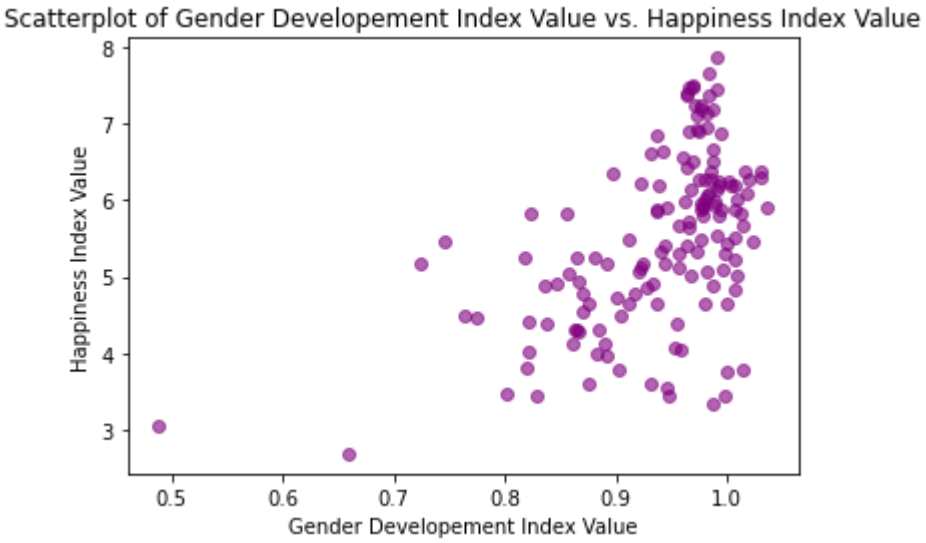
68	Kazakhstan	0.980	77.7	69.2	6.008	1.954113e+05
69	Kenya	0.937	69.0	64.3	4.656	1.087397e+05
70	South Korea	0.936	86.0	79.9	5.840	1.723546e+06
71	Kuwait	0.983	76.6	74.8	6.094	1.318257e+05
72	Kyrgyzstan	0.957	75.6	67.4	5.297	8.676333e+03
73	Laos	0.927	69.7	66.1	4.859	1.805300e+04
74	Latvia	1.036	80.0	70.2	5.901	3.772267e+04
75	Lebanon	0.892	80.9	77.1	5.167	3.537233e+04
76	Lesotho	1.014	57.6	51.2	3.795	2.433000e+03
77	Liberia	0.890	65.5	62.7	4.135	3.289333e+03
78	Libya	0.976	76.0	70.1	5.494	3.728967e+04
79	Lithuania	1.030	81.4	70.3	6.309	6.336067e+04
80	Luxembourg	0.976	84.3	80.2	7.243	8.073933e+04
81	Madagascar	0.952	68.7	65.4	4.071	1.425033e+04
82	Malawi	0.986	67.4	61.1	3.335	1.198100e+04
83	Malaysia	0.972	78.3	74.2	5.339	3.811413e+05
84	Mali	0.821	60.1	58.5	4.416	1.830333e+04
85	Malta	0.966	84.3	80.7	6.910	1.641900e+04
86	Mauritania	0.864	66.5	63.3	4.314	8.745000e+03
87	Mauritius	0.976	78.5	71.7	5.882	1.119167e+04
88	Mexico	0.960	77.9	72.2	6.550	1.263670e+06
89	Moldova	1.014	76.2	67.6	5.682	1.321367e+04
90	Mongolia	1.023	74.1	65.8	5.465	1.465167e+04
91	Montenegro	0.966	79.3	74.4	5.650	5.575000e+03
92	Morocco	0.835	77.9	75.4	4.897	1.301077e+05
93	Mozambique	0.912	63.7	57.8	4.654	1.599933e+04
94	Namibia	1.007	66.5	60.7	4.834	1.181133e+04
95	Nepal	0.933	72.2	69.3	4.910	3.613200e+04
96	Netherlands	0.966	84.0	80.6	7.463	9.741517e+05
97	New Zealand	0.964	84.0	80.6	7.370	2.349120e+05

98	Nicaragua	1.012	78.0	70.9	5.819	1.410967e+04
99	Niger	0.724	63.6	61.3	5.164	1.444367e+04
100	Nigeria	0.881	55.6	53.8	5.252	4.582930e+05
101	Norway	0.990	84.4	80.4	7.444	4.498873e+05
102	Oman	0.936	80.3	76.1	6.853	8.606867e+04
103	Pakistan	0.745	68.3	66.3	5.472	3.268883e+05
104	Palestine	0.870	75.8	72.4	4.554	1.776533e+04
105	Panama	1.019	81.8	75.4	6.281	6.254267e+04
106	Paraguay	0.966	76.4	72.3	5.713	3.871533e+04
107	Peru	0.957	79.5	74.1	5.680	2.219263e+05
108	Philippines	1.007	75.5	67.3	5.869	3.857457e+05
109	Poland	1.007	82.6	74.8	6.201	6.623237e+05
110	Portugal	0.988	84.9	79.0	5.920	2.447600e+05
111	Qatar	1.030	82.0	79.1	6.375	1.824470e+05
112	Romania	0.991	79.5	72.6	6.151	2.775630e+05
113	Russia	1.007	77.8	67.1	5.514	1.797463e+06
114	Rwanda	0.945	71.1	66.8	3.561	1.116667e+04
115	Saudi Arabia	0.896	76.8	73.9	6.356	8.480823e+05
116	Senegal	0.870	69.9	65.8	4.769	2.652633e+04
117	Serbia	0.977	78.6	73.4	5.936	5.970800e+04
118	Sierra Leone	0.884	55.5	53.9	4.306	4.029333e+03
119	Singapore	0.985	85.7	81.5	6.375	3.868690e+05
120	Slovakia	0.992	81.0	74.0	6.235	1.108207e+05
121	Slovenia	1.001	84.0	78.6	6.249	5.910233e+04
122	South Africa	0.986	67.7	60.7	4.884	3.778557e+05
123	Spain	0.986	86.2	80.8	6.513	1.365563e+06
124	Sri Lanka	0.955	80.3	73.6	4.400	7.964500e+04
125	Sudan	0.860	67.2	63.5	4.139	4.638167e+04
126	Suriname	0.985	75.1	68.5	6.269	3.331000e+03
127	Sweden	0.983	84.6	81.0	7.375	5.908080e+05

```
128              Switzerland      0.968             85.6             81.9      7.509
7.908443e+05
129                    Syria      0.829             78.1             67.9      3.462
1.850900e+04
130               Tajikistan      0.823             73.4             68.9      5.829
8.908667e+03
131                 Tanzania      0.948             67.2             63.6      3.445
6.969900e+04
132                 Thailand      1.008             80.9             73.5      6.012
5.141783e+05
133                     Togo      0.822             61.9             60.2      4.023
7.972667e+03
134      Trinidad and Tobago      1.003             76.2             70.9      6.192
2.404067e+04
135                  Tunisia      0.900             78.7             74.7      4.741
4.411333e+04
136                   Turkey      0.924             80.6             74.7      5.186
7.962857e+05
137                   Uganda      0.863             65.6             61.0      4.322
4.249633e+04
138                  Ukraine      1.000             76.8             67.1      4.662
1.851290e+05
139     United Arab Emirates      0.931             79.3             77.3      6.604
4.072170e+05
140           United Kingdom      0.970             83.0             79.6      7.233
3.049843e+06
141            United States      0.994             81.4             76.3      6.883
2.297500e+07
142                  Uruguay      1.016             81.5             74.1      6.372
6.137000e+04
143               Uzbekistan      0.939             73.8             69.6      6.205
6.868767e+04
144                Venezuela      1.009             76.0             68.3      5.006
2.236210e+05
145                  Vietnam      0.997             79.5             71.3      5.296
3.492013e+05
146                    Yemen      0.488             67.8             64.4      3.058
2.553800e+04
147                   Zambia      0.958             66.9             60.8      4.041
2.211300e+04
148                 Zimbabwe      0.931             62.9             59.8      3.616
2.876167e+04
```

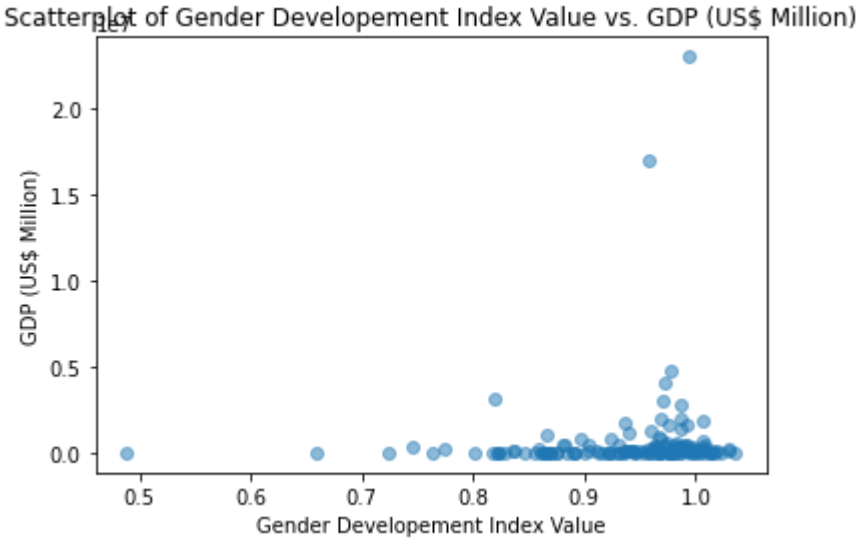
In [66]:

```
plt.scatter(df['GDI_Value'], df['Happiness'], alpha=0.6, color='purple')
plt.xlabel("Gender Developement Index Value")
plt.ylabel("Happiness Index Value")
plt.title('Scatterplot of Gender Developement Index Value vs. Happiness Index Value')
plt.show()
```



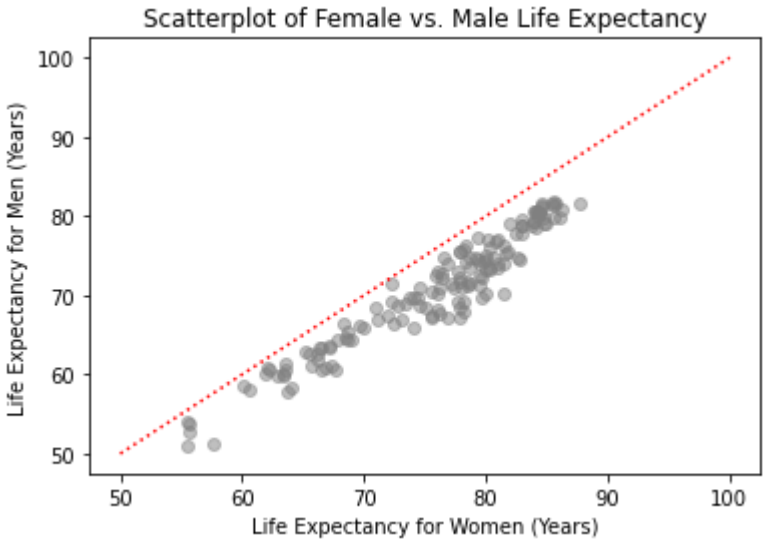
In [67]:

```
plt.scatter(df['GDI_Value'], df['Mean_GDP'], alpha=0.5)
plt.xlabel("Gender Developement Index Value")
plt.ylabel("GDP (US$ Million)")
plt.title('Scatterplot of Gender Developement Index Value vs. GDP (US$ Million)')
plt.show()
```



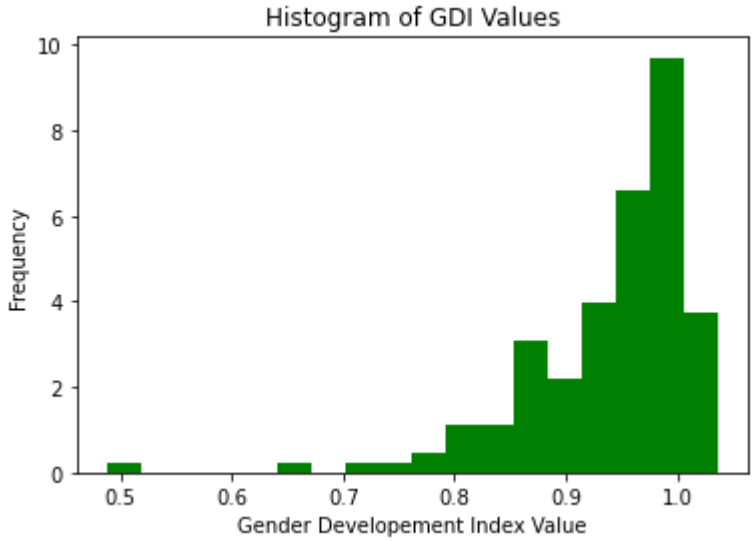
In [68]:

```
plt.scatter(df['Lif_Expec_Female'], df['Lif_Excep_Male'], alpha=0.5, color='gray')
plt.xlabel("Life Expectancy for Women (Years)")
plt.ylabel("Life Expectancy for Men (Years)")
x = np.linspace(50, 100, 1000)
plt.plot(x, x, color='red', linestyle=':')
plt.title('Scatterplot of Female vs. Male Life Expectancy')
plt.show()
```



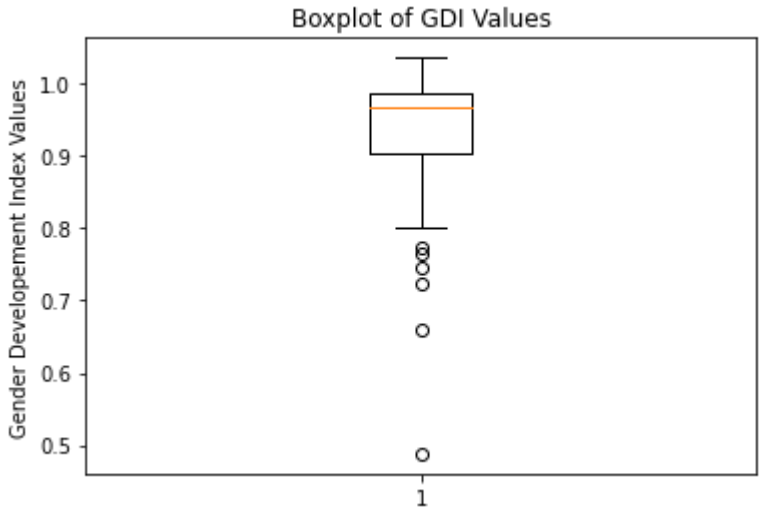
In [69]:

```
plt.hist(df['GDI_Value'], bins='auto', density=True, color='green')
plt.xlabel('Gender Developement Index Value')
plt.ylabel('Frequency')
plt.title('Histogram of GDI Values')
plt.show()
```



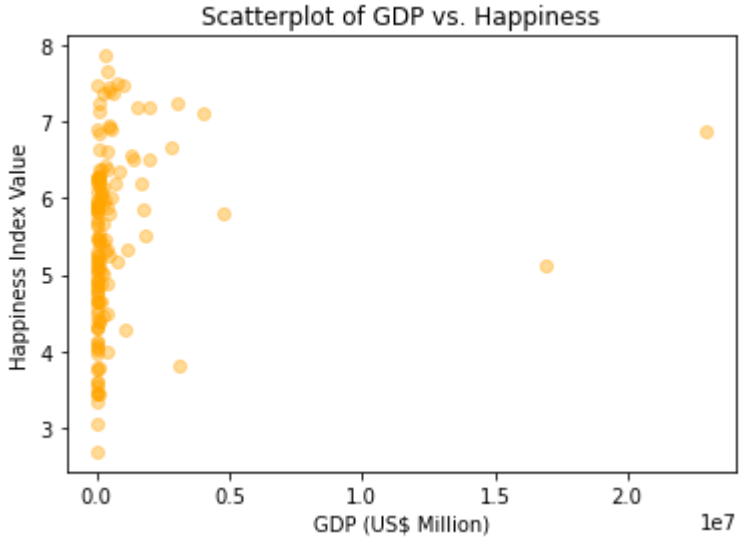
In [70]:

```
plt.boxplot(df['GDI_Value'])
plt.ylabel('Gender Developement Index Values')
plt.title('Boxplot of GDI Values')
plt.show()
```



In [71]:

```
plt.scatter(df['Mean_GDP'], df['Happiness'], alpha=0.4, color='orange')
plt.xlabel("GDP (US$ Million)")
plt.ylabel("Happiness Index Value")
plt.title('Scatterplot of GDP vs. Happiness')
plt.show()
```



Summary

In order to complete this project, I had to learn to do a handful of new things, and I was able to practice some skills that had been introduced to me in other courses. Starting with the things that are completely new to me, I pulled data from an API for the first time when learning to do this project. I also performed web scraping on a table from an html website for the first time. Once I had gotten my data, I was able to use my python skills to clean the data, like I had been taught to do in previous courses. However, once the dataframes were ready I had to learn how to convert them to tables inside an SQL database. I also ended up learning how to join tables inside databases, how to do string manipulation methods with SQL, and how to convert from a table in an SQL database back to a pandas dataframe.

For ethical implications, the most blatant ethical issue comes from the fact that I am using population data divided along country lines. Every country constitutes a variety of peoples, and it may be unethical to draw arbitrary boundaries around these groups and then generalize about their gender equality or general happiness. Furthermore, I removed all time data related to the country metrics, and I assumed that gender equality data from 2019 will be relevant to the GDP of countries from 2019-2021. I also assumed that GDP will be generally stable year over year and that if 2021 data was missing for the GDP of a country, that it could be substituted with 2020 or 2019 data. The last ethical consideration I would make is that there are three different international organizations which are performing the GDP estimates. Each of these organizations has their own agendas and priorities about which territories are considered independent countries, and what their GDP should be, which could have influenced their estimates.

In [ ]: