

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

```
In [267]: # Load Data
csvdf = pd.read_csv("C:/Users/nneam/OneDrive/Documents/540Assignments/world_happiness.csv")
```

```
In [268]: pd.set_option('display.max_rows', csvdf.shape[0]+1)
```

```
In [269]: # Preview Data
csvdf.head(300)
```

Out[269]:

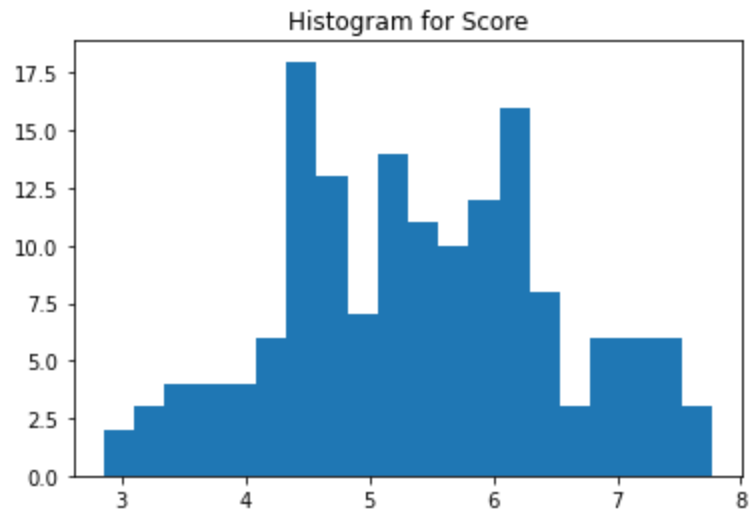
	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	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
5	6	Switzerland	7.480	1.452	1.526	1.052	0.572	0.263	0.343
6	7	Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373
7	8	New Zealand	7.307	1.303	1.557	1.026	0.585	0.330	0.380
8	9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308
9	10	Austria	7.246	1.376	1.475	1.016	0.532	0.244	0.226
10	11	Australia	7.228	1.372	1.548	1.036	0.557	0.332	0.290

```
In [270]: #Changing country names to match format
csvdf['Country or region'] = csvdf['Country or region'].replace(['Trinidad & Tobago'],'Trinidad and Tobago')
csvdf['Country or region'] = csvdf['Country or region'].replace(['Congo (Brazzaville)'],'Congo')
csvdf['Country or region'] = csvdf['Country or region'].replace(['Congo (Kinshasa)'],'DR Congo')
```

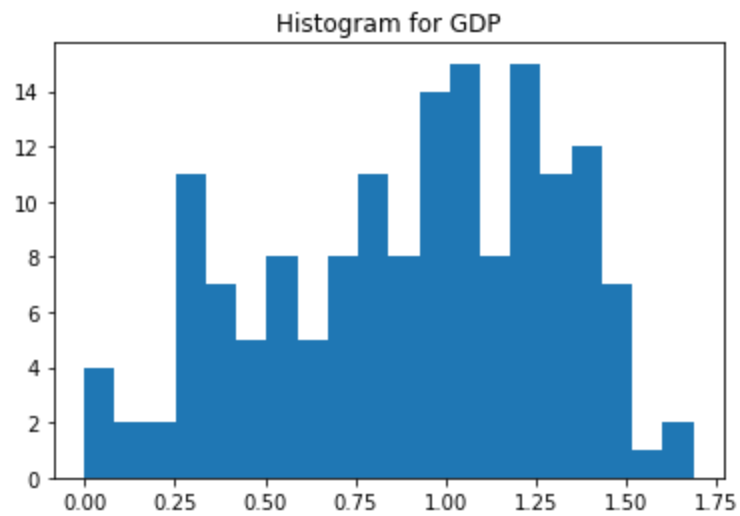
```
In [271]: # Changing header names to program-friendly format
header_names=['Rank', 'Country', 'Score', 'GDP', 'Social_Support', 'Health', 'Freedom', 'Generosity', 'Corruption']
csvdf1 = pd.read_csv("C:/Users/nneam/OneDrive/Documents/540Assignments/world_happiness.csv", header=None, skiprows=1,
```

# Check For Outliers

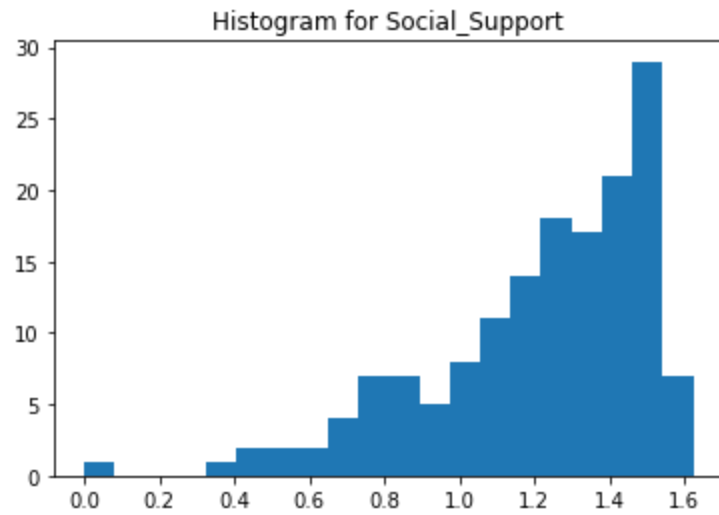
```
In [272]: ▶ plt.title("Histogram for Score")  
plt.hist(csvdf1['Score'],bins=20)  
plt.show()
```



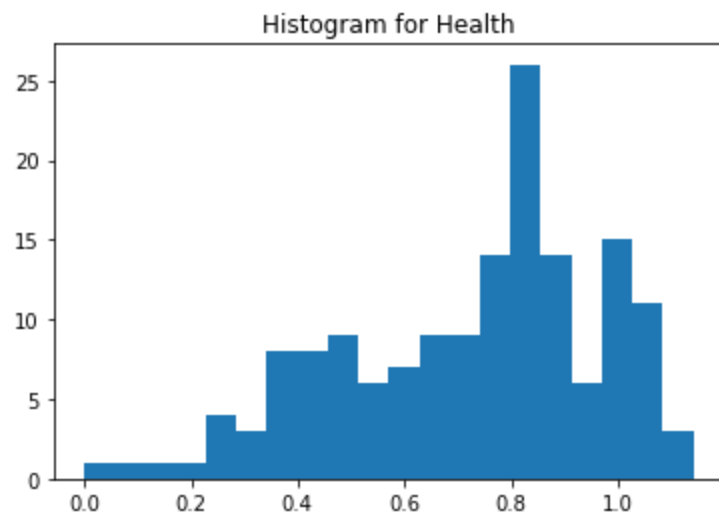
```
In [273]: ▶ plt.title("Histogram for GDP")  
plt.hist(csvdf1['GDP'],bins=20)  
plt.show()
```



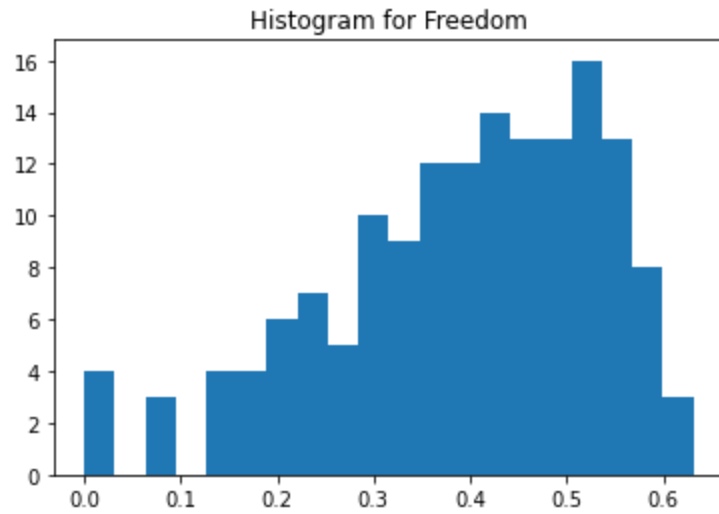
```
In [274]: ▶ plt.title("Histogram for Social_Support")  
plt.hist(csvdf1['Social_Support'],bins=20)  
plt.show()
```



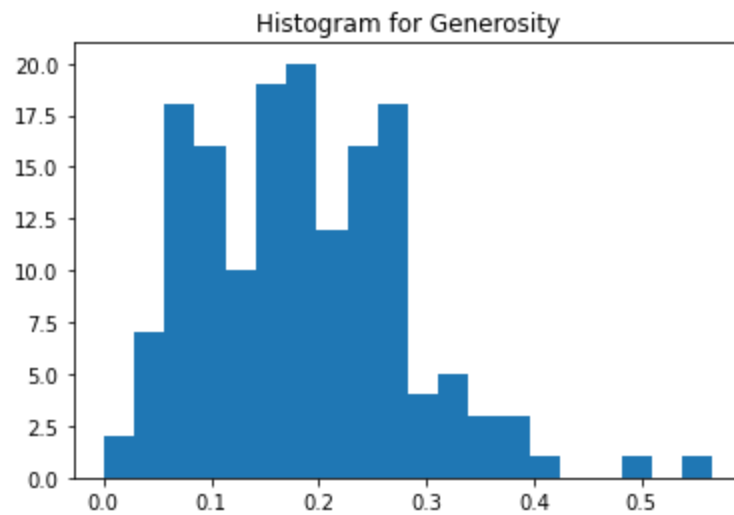
```
In [275]: ▶ plt.title("Histogram for Health")  
plt.hist(csvdf1['Health'],bins=20)  
plt.show()
```



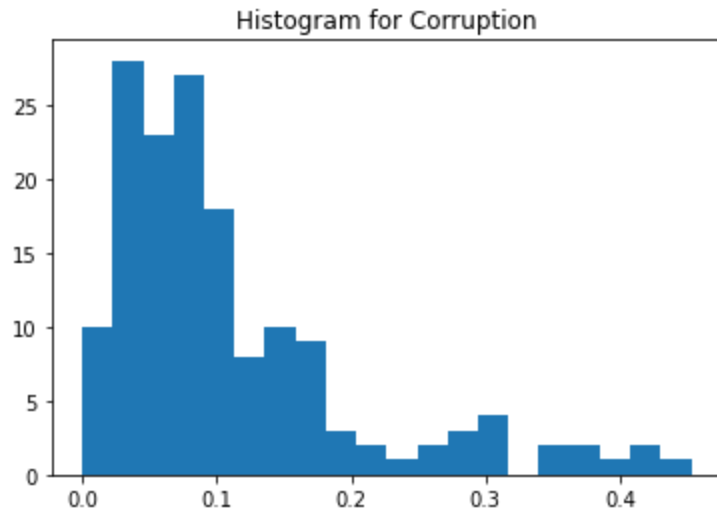
```
In [276]: ▶ plt.title("Histogram for Freedom")  
plt.hist(csvdf1['Freedom'],bins=20)  
plt.show()
```



```
In [277]: ▶ plt.title("Histogram for Generosity")  
plt.hist(csvdf1['Generosity'],bins=20)  
plt.show()
```



```
In [278]: ▶ plt.title("Histogram for Corruption")
plt.hist(csvdf1['Corruption'],bins=20)
plt.show()
```



## Check For Duplicate Countries

```
In [279]: ▶ # Calculating duplicate country values
csvdf1['Country'].duplicated().sum()
# There are 0 duplicates
```

Out[279]: 0

## Fix casing or inconsistent values

```
In [280]: # Checking for missing values  
csvdf1.isnull().sum()
```

```
Out[280]: Rank          0  
Country          0  
Score            0  
GDP              0  
Social_Support   0  
Health           0  
Freedom          0  
Generosity       0  
Corruption       0  
dtype: int64
```

## Put Data in More Readable Format

```
In [281]: # def decimals(x):  
        try:  
            return round(float(x), 2)  
        except Exception:  
            return x  
  
        for value in csvdf1.columns:  
            csvdf1[value] = csvdf1[value].map(decimals)
```

```
In [282]: # New Dataframe  
csvdf1.head(5)
```

```
Out[282]:
```

	Rank	Country	Score	GDP	Social_Support	Health	Freedom	Generosity	Corruption
0	1.0	Finland	7.77	1.34	1.59	0.99	0.60	0.15	0.39
1	2.0	Denmark	7.60	1.38	1.57	1.00	0.59	0.25	0.41
2	3.0	Norway	7.55	1.49	1.58	1.03	0.60	0.27	0.34
3	4.0	Iceland	7.49	1.38	1.62	1.03	0.59	0.35	0.12
4	5.0	Netherlands	7.49	1.40	1.52	1.00	0.56	0.32	0.30

```
In [283]: # Dropping Rank column as it's unnecessary for this project  
csvdf2 = csvdf1.drop(["Rank"], axis = 1, inplace = True)
```

```
In [285]: ▶ # Final Dataset  
csvdf1.head(5)
```

Out[285]:

	Country	Score	GDP	Social_Support	Health	Freedom	Generosity	Corruption
0	Finland	7.77	1.34	1.59	0.99	0.60	0.15	0.39
1	Denmark	7.60	1.38	1.57	1.00	0.59	0.25	0.41
2	Norway	7.55	1.49	1.58	1.03	0.60	0.27	0.34
3	Iceland	7.49	1.38	1.62	1.03	0.59	0.35	0.12
4	Netherlands	7.49	1.40	1.52	1.00	0.56	0.32	0.30

## Website

```
In [226]: ▶ import bs4 as bs  
import urllib.request
```

```
In [227]: ▶ #Assign URL  
x = 'https://www.disabled-world.com/calculators-charts/wpc.php'
```

```
In [228]: ▶ # Pull data and apply headers  
y = pd.read_html(x, header = 1)
```

```
In [229]: ▶ #Pull second table on page  
webdf2 = y[1]
```

```
In [230]:  # Display dataframe
webdf2.head(50)
```

Out[230]:

	Rank	Country	Population	Density (P/Km²)	Med. Age	World Share
0	1	China	1388232693	148	37	0.185
1	2	India(Indian Life Expectancy by States)	1342512706	452	27	0.179
2	3	United States(Population of American Cities an...)	326474013	36	38	0.043
3	4	Indonesia	263510146	146	28	0.035
4	5	Brazil	211243220	25	31	0.028
5	6	Pakistan	196744376	255	23	0.026
6	7	Nigeria	191835936	211	18	0.026
7	8	Bangladesh	164827718	1266	26	0.022
8	9	Russia	143375006	9	39	0.019
9	10	Mexico	130222815	67	27	0.017
10	11	Japan	126045211	346	47	0.017
11	12	Ethiopia	104344901	104	19	0.014
12	13	Philippines	103796832	348	24	0.014
13	14	Viet Nam	95414640	308	30	0.013
14	15	Egypt	95215102	96	25	0.013
15	16	DR Congo	82242685	36	17	0.011
16	17	Iran	80945718	50	30	0.011
17	18	Germany	80636124	231	46	0.011
18	19	Turkey	80417526	105	30	0.011
19	20	Thailand	68297547	134	38	0.009
20	21	U.K.	65511098	271	40	0.009
21	22	France	64938716	119	41	0.009
22	23	Italy	59797978	203	46	0.008
23	24	Tanzania	56877529	64	17	0.008
24	25	South Africa	55436360	46	26	0.007
25	26	Myanmar	54836483	84	28	0.007
26	27	South Korea	50704971	522	41	0.007



Rank		Country	Population	Density (P/Km²)	Med. Age	World Share
27	28	Colombia	49067981	44	30	0.007
28	29	Kenya	48466928	85	19	0.006
29	30	Spain	46070146	92	43	0.006
30	31	Ukraine	44405055	77	40	0.006
31	32	Argentina	44272125	16	31	0.006
32	33	Sudan	42166323	24	19	0.006
33	34	Uganda	41652938	209	16	0.006
34	35	Algeria	41063753	17	28	0.005
35	36	Iraq	38654287	89	19	0.005
36	37	Poland	38563573	126	40	0.005
37	38	Canada(Life Expectancy by Canadian Province an...	36626083	4	41	0.005
38	39	Morocco	35241418	79	28	0.005
39	40	Afghanistan	34169169	52	18	0.005
40	41	Saudi Arabia	32742664	15	28	0.004
41	42	Peru	32166473	25	28	0.004
42	43	Venezuela	31925705	36	27	0.004
43	44	Malaysia	31164177	95	29	0.004
44	45	Uzbekistan	30690914	72	26	0.004
45	46	Mozambique	29537914	38	17	0.004
46	47	Nepal	29187037	204	23	0.004
47	48	Ghana	28656723	126	21	0.004
48	49	Yemen	28119546	53	19	0.004
49	50	Angola	26655513	21	16	0.004

In [231]: **#Dropping unnecessary text in country name**  
webdf2['Country'] = webdf2['Country'].replace(['Canada(Life Expectancy by Canadian Province and Territory)'], 'Canada')  
webdf2['Country'] = webdf2['Country'].replace(['United States(Population of American Cities and Towns)(Life Expectancy by State)'], 'United States')  
webdf2['Country'] = webdf2['Country'].replace(['India(Indian Life Expectancy by States)'], 'India')

```
In [232]: ▶ pd.set_option('display.max_rows', webdf2.shape[0]+1)
```

In [233]: webdf2.head(50)

Out[233]:

	Rank	Country	Population	Density (P/Km²)	Med. Age	World Share
0	1	China	1388232693	148	37	0.185
1	2	India	1342512706	452	27	0.179
2	3	United States	326474013	36	38	0.043
3	4	Indonesia	263510146	146	28	0.035
4	5	Brazil	211243220	25	31	0.028
5	6	Pakistan	196744376	255	23	0.026
6	7	Nigeria	191835936	211	18	0.026
7	8	Bangladesh	164827718	1266	26	0.022
8	9	Russia	143375006	9	39	0.019
9	10	Mexico	130222815	67	27	0.017
10	11	Japan	126045211	346	47	0.017
11	12	Ethiopia	104344901	104	19	0.014
12	13	Philippines	103796832	348	24	0.014
13	14	Viet Nam	95414640	308	30	0.013
14	15	Egypt	95215102	96	25	0.013
15	16	DR Congo	82242685	36	17	0.011
16	17	Iran	80945718	50	30	0.011
17	18	Germany	80636124	231	46	0.011
18	19	Turkey	80417526	105	30	0.011
19	20	Thailand	68297547	134	38	0.009
20	21	U.K.	65511098	271	40	0.009
21	22	France	64938716	119	41	0.009
22	23	Italy	59797978	203	46	0.008
23	24	Tanzania	56877529	64	17	0.008
24	25	South Africa	55436360	46	26	0.007
25	26	Myanmar	54836483	84	28	0.007
26	27	South Korea	50704971	522	41	0.007
27	28	Colombia	49067981	44	30	0.007

	Rank	Country	Population	Density (P/Km²)	Med. Age	World Share
28	29	Kenya	48466928	85	19	0.006
29	30	Spain	46070146	92	43	0.006
30	31	Ukraine	44405055	77	40	0.006
31	32	Argentina	44272125	16	31	0.006
32	33	Sudan	42166323	24	19	0.006
33	34	Uganda	41652938	209	16	0.006
34	35	Algeria	41063753	17	28	0.005
35	36	Iraq	38654287	89	19	0.005
36	37	Poland	38563573	126	40	0.005
37	38	Canada	36626083	4	41	0.005
38	39	Morocco	35241418	79	28	0.005
39	40	Afghanistan	34169169	52	18	0.005
40	41	Saudi Arabia	32742664	15	28	0.004
41	42	Peru	32166473	25	28	0.004
42	43	Venezuela	31925705	36	27	0.004
43	44	Malaysia	31164177	95	29	0.004
44	45	Uzbekistan	30690914	72	26	0.004
45	46	Mozambique	29537914	38	17	0.004
46	47	Nepal	29187037	204	23	0.004
47	48	Ghana	28656723	126	21	0.004
48	49	Yemen	28119546	53	19	0.004
49	50	Angola	26655513	21	16	0.004

## Find Duplicates Values

```
In [234]: webdf2['Country'].duplicated().sum()
# There are 0 duplicates
```

Out[234]: 0

## Remove Unnecessary Columns

```
In [235]: webdf3=webdf2.copy()
webdf3.reset_index(inplace=True)
```

```
In [236]: #Dropping "Rank" column
webdf3 = webdf3.drop(columns=['Density (P/Km²)'])
webdf3 = webdf3.drop(columns=['Rank'])
```

```
In [237]: #Display new df
webdf3.head(10)
```

Out[237]:

	index	Country	Population	Med. Age	World Share
0	0	China	1388232693	37	0.185
1	1	India	1342512706	27	0.179
2	2	United States	326474013	38	0.043
3	3	Indonesia	263510146	28	0.035
4	4	Brazil	211243220	31	0.028
5	5	Pakistan	196744376	23	0.026
6	6	Nigeria	191835936	18	0.026
7	7	Bangladesh	164827718	26	0.022
8	8	Russia	143375006	39	0.019
9	9	Mexico	130222815	27	0.017

```
In [238]: webdf3 = webdf3.drop(columns=['index'])
```

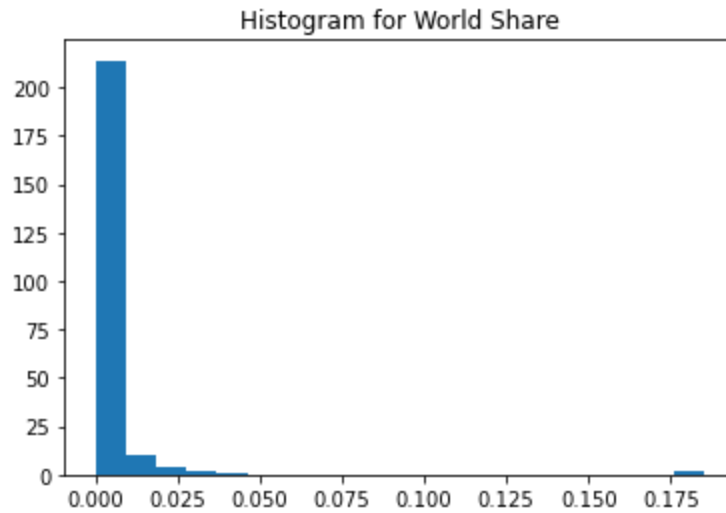
## Identify outliers and bad data

```
In [239]: # Import Library
import matplotlib.pyplot as plt
```

```
In [240]: plt.title("Histogram for Age")
plt.hist(webdf3['Med. Age'],bins=40)
plt.show()
```



```
In [241]: ▶ plt.title("Histogram for World Share")
plt.hist(webdf3['World Share'],bins=20)
plt.show()
```



## Format data into a more readable format


```
In [242]: ▶ #Make population columns easier to read
webdf3['Population'] = webdf3['Population'].map("{:,}".format)
```

```
In [243]:  # Display Change  
webdf3.head(10)
```

Out[243]:


	Country	Population	Med. Age	World Share
0	China	1,388,232,693	37	0.185
1	India	1,342,512,706	27	0.179
2	United States	326,474,013	38	0.043
3	Indonesia	263,510,146	28	0.035
4	Brazil	211,243,220	31	0.028
5	Pakistan	196,744,376	23	0.026
6	Nigeria	191,835,936	18	0.026
7	Bangladesh	164,827,718	26	0.022
8	Russia	143,375,006	39	0.019
9	Mexico	130,222,815	27	0.017

## Fix casing or inconsistent values

```
In [244]:  # Checking for missing values  
webdf3.isnull().sum()
```

Out[244]: Country 0  
Population 0  
Med. Age 0  
World Share 0  
dtype: int64

## API Data

```
In [108]:  # connecting to world bank data api  
import wpdata  
import pandas as pd
```



```
In [37]: wldata.get_source()
```

```
Out[37]:
```

id	name
1	Doing Business
2	World Development Indicators
3	Worldwide Governance Indicators
5	Subnational Malnutrition Database
6	International Debt Statistics
11	Africa Development Indicators
12	Education Statistics
13	Enterprise Surveys
14	Gender Statistics
15	Global Economic Monitor
16	Health Nutrition and Population Statistics
18	IDA Results Measurement System
19	Millennium Development Goals
20	Quarterly Public Sector Debt
22	Quarterly External Debt Statistics SDDS
23	Quarterly External Debt Statistics GDDS
24	Poverty and Equity
25	Jobs
27	Global Economic Prospects
28	Global Financial Inclusion
29	The Atlas of Social Protection: Indicators of Resilience and Equity
30	Exporter Dynamics Database – Indicators at Country-Year Level
31	Country Policy and Institutional Assessment
32	Global Financial Development
33	G20 Financial Inclusion Indicators
34	Global Partnership for Education
35	Sustainable Energy for All
36	Statistical Capacity Indicators
37	LAC Equity Lab
38	Subnational Poverty
39	Health Nutrition and Population Statistics by Wealth Quintile
40	Population estimates and projections
41	Country Partnership Strategy for India (FY2013 - 17)
43	Adjusted Net Savings
44	Readiness for Investment in Sustainable Energy
45	Indonesia Database for Policy and Economic Research
46	Sustainable Development Goals
50	Subnational Population
54	Joint External Debt Hub
57	WDI Database Archives
58	Universal Health Coverage
59	Wealth Accounts
60	Economic Fitness

```

61 PPPs Regulatory Quality
62 International Comparison Program (ICP) 2011
63 Human Capital Index
64 Worldwide Bureaucracy Indicators
65 Health Equity and Financial Protection Indicators
66 Logistics Performance Index
67 PEFA 2011
68 PEFA 2016
69 Global Financial Inclusion and Consumer Protection Survey
70 Economic Fitness 2
71 International Comparison Program (ICP) 2005
72 PEFA_Test
73 Global Financial Inclusion and Consumer Protection Survey (Internal)
75 Environment, Social and Governance (ESG) Data
76 Remittance Prices Worldwide (Sending Countries)
77 Remittance Prices Worldwide (Receiving Countries)
78 ICP 2017
79 PEFA_GRPFM
80 Gender Disaggregated Labor Database (GDLD)
81 International Debt Statistics: DSSI

```

```

In [38]: # Global Financial Development
wbdata.get_indicator(source=2)

```

```

Out[38]: id name
-----
AG.AGR.TRAC.NO Agricultural machinery, tractors
AG.CON.FERT.PT.ZS Fertilizer consumption (% of fertilizer production)
AG.CON.FERT.ZS Fertilizer consumption (kilograms per hectare of arable land)
AG.LND.AGRI.K2 Agricultural land (sq. km)
AG.LND.AGRI.ZS Agricultural land (% of land area)
AG.LND.ARBL.HA Arable land (hectares)
AG.LND.ARBL.HA.PC Arable land (hectares per person)
AG.LND.ARBL.ZS Arable land (% of land area)
AG.LND.CREL.HA Land under cereal production (hectares)
AG.LND.CROP.ZS Permanent cropland (% of land area)
AG.LND.EL5M.RU.K2 Rural land area where elevation is below 5 meters (sq. km)
AG.LND.EL5M.RU.ZS Rural land area where elevation is below 5 meters (% of total land area)
AG.LND.EL5M.UR.K2 Urban land area where elevation is below 5 meters (sq. km)
AG.LND.EL5M.UR.ZS Urban land area where elevation is below 5 meters (% of total land area)
AG.LND.EL5M.ZS Land area where elevation is below 5 meters (% of total land area)
AG.LND.FRST.K2 Forest area (sq. km)
AG.LND.FRST.ZS Forest area (% of land area)

```

In [39]: *#Select Indicators*

```
Ind = {"NE.EXP.GNFS.ZS" : "Exports of goods and services (% of GDP)",
       "SP.RUR.TOTL" : "Rural Population",
       "SP.URB.TOTL": "Urban Population",
       "SL.TLF.TOTL.IN" : 'Labor Force',
       "SP.DYN.IMRT.IN": 'Mortality Rate, Infant (Per 1000)',
       "SP.DYN.LE00.IN" : 'Life Expectancy'
}
```

In [40]: *#Assigning indicators to a variable*

```
indicators = Ind
```

In [352]: *#Set timeframe*

```
import datetime
dd = datetime.datetime(2018, 1, 1), datetime.datetime(2018, 12, 31)
```

In [353]: *#Creating DF*

```
apidf=wbddata.get_dataframe(indicators,country = 'all', data_date= dd, freq='Y')
```

In [354]: *# Display data info*

```
apidf.info()
```

```
<class 'wbddata.api.WBDataFrame'>
```

```
Index: 264 entries, Arab World to Zimbabwe
```

```
Data columns (total 6 columns):
```

#	Column	Non-Null Count	Dtype
0	Exports of goods and services (% of GDP)	216 non-null	float64
1	Rural Population	260 non-null	float64
2	Urban Population	260 non-null	float64
3	Labor Force	232 non-null	float64
4	Mortality Rate, Infant (Per 1000)	239 non-null	float64
5	Life Expectancy	244 non-null	float64

```
dtypes: float64(6)
```

```
memory usage: 24.4+ KB
```


In [355]:

#Display Data  
apidf.tail(40)

Out[355]:

	Exports of goods and services (% of GDP)	Rural Population	Urban Population	Labor Force	Mortality Rate, Infant (Per 1000)	Life Expectancy
country						
South Africa	29.907083	19439954.0	38339668.0	22947458.0	27.8	63.857000
South Sudan	NaN	8822993.0	2152927.0	4621451.0	62.4	57.604000
Spain	35.116658	9209330.0	37588424.0	23065037.0	2.6	83.431707
Sri Lanka	22.917932	17666251.0	4003749.0	8893806.0	6.4	76.812000
St. Kitts and Nevis	59.174850	36302.0	16139.0	NaN	13.2	NaN
St. Lucia	NaN	147916.0	33973.0	99309.0	19.7	76.057000
St. Martin (French part)	NaN	NaN	NaN	NaN	NaN	79.870732
St. Vincent and the Grenadines	NaN	52683.0	57527.0	56243.0	13.8	72.415000
Sudan	10.248835	27320646.0	14480887.0	12055376.0	42.0	65.095000
Suriname	NaN	195491.0	380500.0	214906.0	16.6	71.570000
Sweden	45.683062	1278923.0	8896291.0	5411383.0	2.2	82.558537
Switzerland	66.129421	2231010.0	6283319.0	4953967.0	3.6	83.753659
Syrian Arab Republic	NaN	7749502.0	9156781.0	5129679.0	18.0	71.779000
Tajikistan	NaN	6631416.0	2469421.0	2412933.0	30.4	70.879000
Tanzania	NaN	37296263.0	19022085.0	26304005.0	37.0	65.015000
Thailand	64.868885	34749671.0	34678853.0	38907795.0	8.1	76.931000
Timor-Leste	2.740391	880252.0	387720.0	531648.0	39.2	69.260000
Togo	31.316617	4599184.0	3289910.0	3596555.0	47.0	60.760000
Tonga	22.297907	79327.0	23870.0	39833.0	14.4	70.801000
Trinidad and Tobago	NaN	650676.0	739182.0	667593.0	16.1	73.380000
Tunisia	49.041684	3591574.0	7973630.0	4061682.0	14.6	76.505000
Turkey	29.530104	20462214.0	61857510.0	32826049.0	9.2	77.437000
Turkmenistan	22.666195	2832249.0	3018659.0	2614576.0	36.6	68.073000
Turks and Caicos Islands	NaN	2600.0	35065.0	NaN	NaN	NaN
Tuvalu	NaN	4329.0	7179.0	NaN	20.9	NaN

	Exports of goods and services (% of GDP)	Rural Population	Urban Population	Labor Force	Mortality Rate, Infant (Per 1000)	Life Expectancy
country						
Uganda	15.159794	32566140.0	10156999.0	15935453.0	34.6	62.973000
Ukraine	45.199315	13675909.0	30946609.0	20432109.0	7.4	71.582683
United Arab Emirates	93.048123	1298061.0	8332898.0	6752973.0	6.5	77.814000
United Kingdom	30.614969	11033746.0	55426598.0	34329233.0	3.7	81.256098
United States	12.227795	57967430.0	268720071.0	165483017.0	5.6	78.539024
Uruguay	21.004266	160944.0	3288355.0	1749881.0	6.5	77.770000
Uzbekistan	28.042253	16320520.0	16635580.0	15289417.0	16.5	71.573000
Vanuatu	NaN	218708.0	73972.0	125122.0	22.4	70.323000
Venezuela, RB	NaN	3404373.0	25465822.0	12554309.0	21.0	72.128000
Vietnam	105.831805	61223241.0	34317154.0	56915235.0	16.3	75.317000
Virgin Islands (U.S.)	NaN	4578.0	102399.0	52142.0	NaN	79.568293
West Bank and Gaza	19.865352	1089088.0	3479999.0	1214123.0	17.1	73.895000
Yemen, Rep.	NaN	18056198.0	10442489.0	6545429.0	43.6	66.096000
Zambia	36.983166	9800136.0	7551686.0	7134980.0	43.4	63.510000
Zimbabwe	19.000609	9788355.0	4650663.0	6907202.0	39.3	61.195000

In [356]:  *#Copying dataframe to allow country and year to be added as columns*  
 apidf2=apidf.copy()  
 apidf2.reset\_index(inplace=True)

```
In [357]: apidf2.info()
```

```
<class 'wbdata.api.WBDataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 7 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   country                               264 non-null    object
1   Exports of goods and services (% of GDP) 216 non-null    float64
2   Rural Population                        260 non-null    float64
3   Urban Population                        260 non-null    float64
4   Labor Force                            232 non-null    float64
5   Mortality Rate, Infant (Per 1000)        239 non-null    float64
6   Life Expectancy                         244 non-null    float64
dtypes: float64(6), object(1)
memory usage: 14.6+ KB
```

```
In [358]: apidf2.tail(10)
```

Out[358]:

	country	Exports of goods and services (% of GDP)	Rural Population	Urban Population	Labor Force	Mortality Rate, Infant (Per 1000)	Life Expectancy
254	Uruguay	21.004266	160944.0	3288355.0	1749881.0	6.5	77.770000
255	Uzbekistan	28.042253	16320520.0	16635580.0	15289417.0	16.5	71.573000
256	Vanuatu	NaN	218708.0	73972.0	125122.0	22.4	70.323000
257	Venezuela, RB	NaN	3404373.0	25465822.0	12554309.0	21.0	72.128000
258	Vietnam	105.831805	61223241.0	34317154.0	56915235.0	16.3	75.317000
259	Virgin Islands (U.S.)	NaN	4578.0	102399.0	52142.0	NaN	79.568293
260	West Bank and Gaza	19.865352	1089088.0	3479999.0	1214123.0	17.1	73.895000
261	Yemen, Rep.	NaN	18056198.0	10442489.0	6545429.0	43.6	66.096000
262	Zambia	36.983166	9800136.0	7551686.0	7134980.0	43.4	63.510000
263	Zimbabwe	19.000609	9788355.0	4650663.0	6907202.0	39.3	61.195000

In [359]: `apidf2.head(265)`

Out[359]:

	country	Exports of goods and services (% of GDP)	Rural Population	Urban Population	Labor Force	Mortality Rate, Infant (Per 1000)	Life Expectancy
0	Arab World	46.993808	1.724272e+08	2.473634e+08	1.352659e+08	26.606529	71.806825
1	Caribbean small states	NaN	3.591223e+06	3.767742e+06	3.377342e+06	15.019788	73.637108
2	Central Europe and the Baltics	65.131195	3.862182e+07	6.391521e+07	4.972420e+07	4.172168	76.976845
3	Early-demographic dividend	27.120083	1.782322e+09	1.465917e+09	1.281961e+09	28.663155	70.483191
4	East Asia & Pacific	29.487493	9.533289e+08	1.374809e+09	1.263545e+09	12.400000	76.068336
...	...	...	...	...	...	...	...
259	Virgin Islands (U.S.)	NaN	4.578000e+03	1.023990e+05	5.214200e+04	NaN	79.568293
260	West Bank and Gaza	19.865352	1.089088e+06	3.479999e+06	1.214123e+06	17.100000	73.895000
261	Yemen, Rep.	NaN	1.805620e+07	1.044249e+07	6.545429e+06	43.600000	66.096000
262	Zambia	36.983166	9.800136e+06	7.551686e+06	7.134980e+06	43.400000	63.510000
263	Zimbabwe	19.000609	9.788355e+06	4.650663e+06	6.907202e+06	39.300000	61.195000

264 rows × 7 columns

In [360]: `# Remove Values that are not a country`  
`apidf3 = apidf2.drop(apidf2.index[0:141])`

In [362]: `#Change column names`  
`apidf3.columns=["Country", "GDP_%_of_Exports", "Rural_Pop", "Urban_Pop", "Available_Workers", "Infant_Mortality_per_1000"]`

In [363]: `#Need to view whole df in order to match country layout`  
`pd.set_option('display.max_rows', apidf3.shape[0]+1)`

```
In [364]: #Changing countries to match format of website data
apidf3['Country'] = apidf3['Country'].replace(['Congo, Dem. Rep.'],'DR Congo')
apidf3['Country'] = apidf3['Country'].replace(['Congo, Rep.'],'Congo')
apidf3['Country'] = apidf3['Country'].replace(['Egypt, Arab Rep.'],'Egypt')
apidf3['Country'] = apidf3['Country'].replace(['Iran, Islamic Rep.'],'Iran')
apidf3['Country'] = apidf3['Country'].replace(['Korea, Dem. People's Rep.'],'North Korea')
apidf3['Country'] = apidf3['Country'].replace(['Korea, Rep.'],'South Korea')
apidf3['Country'] = apidf3['Country'].replace(['Kyrgyz Republic'],'Kyrgyzstan')
apidf3['Country'] = apidf3['Country'].replace(['Macao SAR, China'],'Macao')
apidf3['Country'] = apidf3['Country'].replace(['Lao PDR'],'Laos')
apidf3['Country'] = apidf3['Country'].replace(['Micronesia, Fed. Sts'],'Micronesia')
apidf3['Country'] = apidf3['Country'].replace(['Russian Federation'],'Russia')
apidf3['Country'] = apidf3['Country'].replace(['Sint Maarten (Dutch part)'],'Sint Maarten')
apidf3['Country'] = apidf3['Country'].replace(['Slovak Republic'],'Slovakia')
apidf3['Country'] = apidf3['Country'].replace(['St. Kitts and Nevis'],'Saint Kitts and Nevis')
apidf3['Country'] = apidf3['Country'].replace(['St. Lucia'],'Saint Lucia')
apidf3['Country'] = apidf3['Country'].replace(['St. Vincent and the Grenadines'],'St. Vincent & Grenadines')
apidf3['Country'] = apidf3['Country'].replace(['Syrian Arab Republic'],'Syria')
apidf3['Country'] = apidf3['Country'].replace(['Venezuela, RB'],'Venezuela')
apidf3['Country'] = apidf3['Country'].replace(['Virgin Islands (U.S.)'],'United States Virgin Islands')
apidf3['Country'] = apidf3['Country'].replace(['Yemen, Rep.'],'Yemen')
```

```
In [365]: apidf3.head(600)
```

Out[365]:

	Country	GDP_%_of_Exports	Rural_Pop	Urban_Pop	Available_Workers	Infant_Mortality_per_1000	Life_Expectancy
141	Isle of Man	NaN	39863.0	44214.0	NaN	NaN	NaN
142	Israel	29.446099	673494.0	8209306.0	4102822.0	3.0	82.802439
143	Italy	31.451744	17861881.0	42559879.0	26034264.0	2.8	83.346341
144	Jamaica	37.957853	1300904.0	1633951.0	1473383.0	12.3	74.368000
145	Japan	18.524909	10608200.0	115920900.0	68358370.0	1.8	84.210976
146	Jordan	35.638746	898132.0	9057879.0	2579658.0	13.8	74.405000
147	Kazakhstan	37.625226	7780670.5	10495828.0	9030838.0	9.2	73.150000
148	Kenya	13.174140	37501479.0	13891531.0	23057935.0	32.8	66.342000
149	Kiribati	9.630327	53224.0	62623.0	NaN	41.3	68.116000
150	North Korea	NaN	9734737.0	15815082.0	16395998.0	13.7	72.095000
151	South Korea	41.630856	9568386.0	42038247.0	28272711.0	2.8	82.626829



```
In [366]: ▶ #Add Column
apidf3['%_of_available_workers'] = (apidf3['Available_Workers'] / (apidf3['Rural_Pop'] + apidf3['Urban_Pop'])* 100)
```

```
In [367]: ▶ #Round Decimals of Specific Columns
apidf3.round({'GDP_%_of_Exports': 2, 'Infant_Mortality_per_1000': 1, 'Life_Expectancy': 1, '%_of_available_workers': 1})
```

Out[367]:

	Country	GDP_%_of_Exports	Rural_Pop	Urban_Pop	Available_Workers	Infant_Mortality_per_1000	Life_Expectancy	%_of_available.
141	Isle of Man	NaN	39863.0	44214.0	NaN	NaN	NaN	
142	Israel	29.45	673494.0	8209306.0	4102822.0	3.0	82.8	
143	Italy	31.45	17861881.0	42559879.0	26034264.0	2.8	83.3	
144	Jamaica	37.96	1300904.0	1633951.0	1473383.0	12.3	74.4	
145	Japan	18.52	10608200.0	115920900.0	68358370.0	1.8	84.2	
146	Jordan	35.64	898132.0	9057879.0	2579658.0	13.8	74.4	
147	Kazakhstan	37.63	7780670.5	10495828.0	9030838.0	9.2	73.2	
148	Kenya	13.17	37501479.0	13891531.0	23057935.0	32.8	66.3	
149	Kiribati	9.63	53224.0	62623.0	NaN	41.3	68.1	
150	North Korea	NaN	9734737.0	15815082.0	16395998.0	13.7	72.1	
151	South Korea	41.63	9568386.0	42038247.0	28272711.0	2.8	82.6	

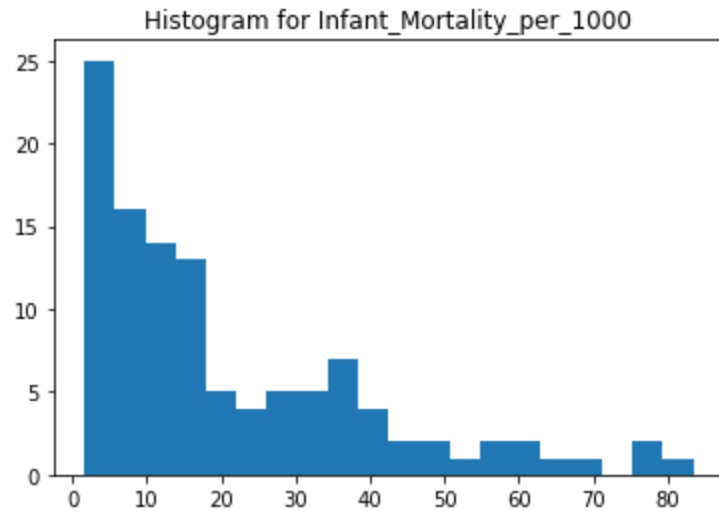
```
In [368]: #Check for outliers  
plt.title("Histogram for Infant_Mortality_per_1000")  
plt.hist(apidf3['Infant_Mortality_per_1000'],bins=20)  
plt.show()
```

C:\Users\nneam\anaconda3\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered in greater\_equal

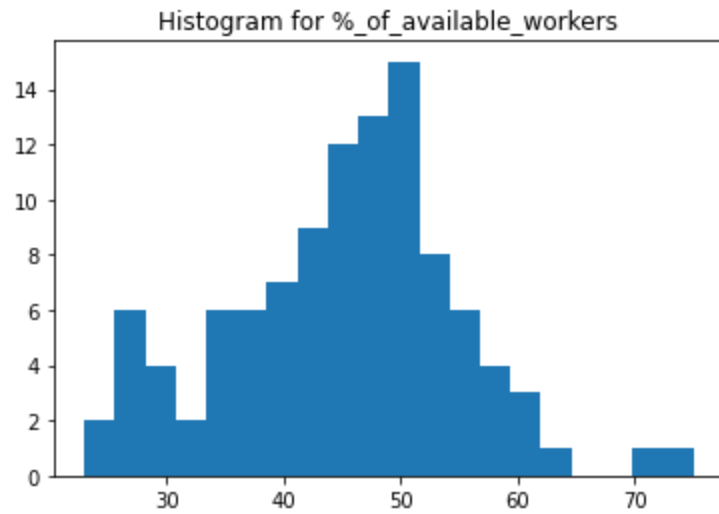
keep = (tmp\_a >= first\_edge)

C:\Users\nneam\anaconda3\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less\_equal

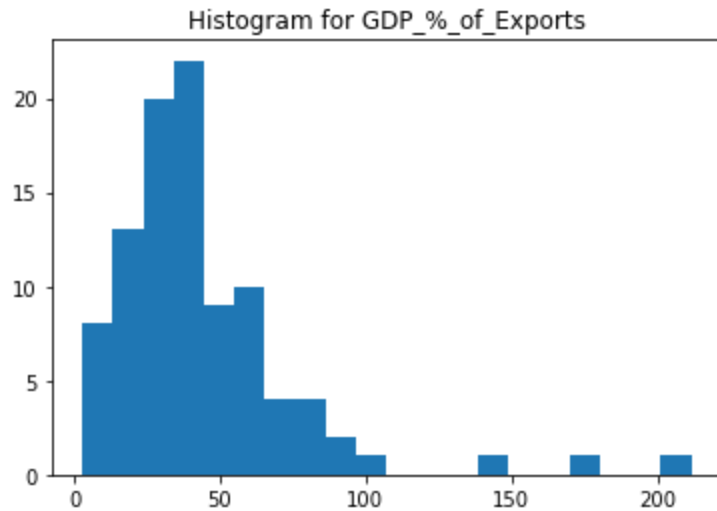
keep &= (tmp\_a <= last\_edge)



```
In [369]: ▶ plt.title("Histogram for %_of_available_workers")  
plt.hist(apidf3['%_of_available_workers'],bins=20)  
plt.show()
```

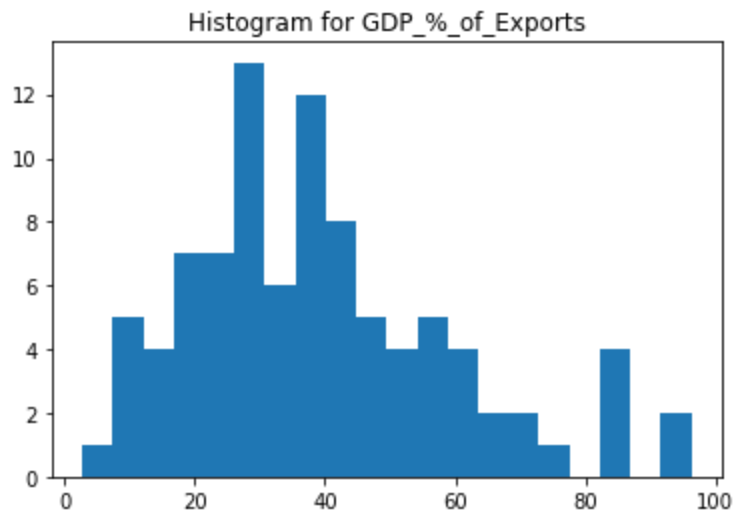


```
In [370]: ▶ plt.title("Histogram for GDP_%_of_Exports")
plt.hist(apidf3['GDP_%_of_Exports'],bins=20)
plt.show()
```



```
In [371]: ▶ #Dropping rows with GDP % over 100%
apidf3.drop(apidf3[apidf3['GDP_%_of_Exports'] > 100 ].index, inplace = True)
```

```
In [372]: ▶ plt.title("Histogram for GDP_%_of_Exports")
plt.hist(apidf3['GDP_%_of_Exports'],bins=20)
plt.show()
```




```
In [373]: ▶ #Assigning final df
apidffinal = apidf3
```

## Final Analysis

```
In [374]: import sqlite3
```

```
In [494]: conn = sqlite3.connect('final-2.db')
```

```
In [376]:  #Setting cursor
          c = conn.cursor()
```

```
In [382]: ▶ csvdf1.to_sql('csvf1', conn)
```

```
In [383]:  #Executing Queryy
c.execute("SELECT * FROM csvf1").fetchall()
```

```
Out[383]: [(0, 'Finland', 7.77, 1.34, 1.59, 0.99, 0.6, 0.15, 0.39),
(1, 'Denmark', 7.6, 1.38, 1.57, 1.0, 0.59, 0.25, 0.41),
(2, 'Norway', 7.55, 1.49, 1.58, 1.03, 0.6, 0.27, 0.34),
(3, 'Iceland', 7.49, 1.38, 1.62, 1.03, 0.59, 0.35, 0.12),
(4, 'Netherlands', 7.49, 1.4, 1.52, 1.0, 0.56, 0.32, 0.3),
(5, 'Switzerland', 7.48, 1.45, 1.53, 1.05, 0.57, 0.26, 0.34),
(6, 'Sweden', 7.34, 1.39, 1.49, 1.01, 0.57, 0.27, 0.37),
(7, 'New Zealand', 7.31, 1.3, 1.56, 1.03, 0.58, 0.33, 0.38),
(8, 'Canada', 7.28, 1.36, 1.5, 1.04, 0.58, 0.28, 0.31),
(9, 'Austria', 7.25, 1.38, 1.48, 1.02, 0.53, 0.24, 0.23),
(10, 'Australia', 7.23, 1.37, 1.55, 1.04, 0.56, 0.33, 0.29),
(11, 'Costa Rica', 7.17, 1.03, 1.44, 0.96, 0.56, 0.14, 0.09),
(12, 'Israel', 7.14, 1.28, 1.46, 1.03, 0.37, 0.26, 0.08),
(13, 'Luxembourg', 7.09, 1.61, 1.48, 1.01, 0.53, 0.19, 0.32),
(14, 'United Kingdom', 7.05, 1.33, 1.54, 1.0, 0.45, 0.35, 0.28),
(15, 'Ireland', 7.02, 1.5, 1.55, 1.0, 0.52, 0.3, 0.31),
(16, 'Germany', 6.99, 1.37, 1.45, 0.99, 0.49, 0.26, 0.27),
(17, 'Belgium', 6.92, 1.36, 1.5, 0.99, 0.47, 0.16, 0.21),
(18, 'United States', 6.89, 1.43, 1.46, 0.87, 0.45, 0.28, 0.13),
(19, 'Japan', 6.85, 1.27, 1.42, 0.82, 0.46, 0.25, 0.21),
(20, 'France', 6.8, 1.3, 1.4, 0.8, 0.4, 0.2, 0.15),
(21, 'Spain', 6.75, 1.25, 1.35, 0.75, 0.35, 0.15, 0.1),
(22, 'Italy', 6.7, 1.2, 1.3, 0.7, 0.3, 0.1, 0.05),
(23, 'South Korea', 6.65, 1.15, 1.25, 0.65, 0.25, 0.05, 0.02),
(24, 'China', 6.6, 1.1, 1.2, 0.6, 0.2, 0.05, 0.01),
(25, 'India', 6.55, 1.05, 1.15, 0.55, 0.15, 0.05, 0.01),
(26, 'Brazil', 6.5, 1.0, 1.1, 0.5, 0.1, 0.05, 0.01),
(27, 'Mexico', 6.45, 0.95, 1.05, 0.45, 0.05, 0.05, 0.01),
(28, 'Russia', 6.4, 0.9, 1.0, 0.4, 0.05, 0.05, 0.01),
(29, 'South Africa', 6.35, 0.85, 0.95, 0.35, 0.05, 0.05, 0.01),
(30, 'Egypt', 6.3, 0.8, 0.9, 0.3, 0.05, 0.05, 0.01),
(31, 'Nigeria', 6.25, 0.75, 0.85, 0.25, 0.05, 0.05, 0.01),
(32, 'Kenya', 6.2, 0.7, 0.8, 0.2, 0.05, 0.05, 0.01),
(33, 'Tanzania', 6.15, 0.65, 0.75, 0.15, 0.05, 0.05, 0.01),
(34, 'Uganda', 6.1, 0.6, 0.7, 0.1, 0.05, 0.05, 0.01),
(35, 'Zambia', 6.05, 0.55, 0.65, 0.05, 0.05, 0.05, 0.01),
(36, 'Zimbabwe', 6.0, 0.5, 0.6, 0.0, 0.05, 0.05, 0.01),
(37, 'Botswana', 5.95, 0.45, 0.55, 0.0, 0.05, 0.05, 0.01),
(38, 'Namibia', 5.9, 0.4, 0.5, 0.0, 0.05, 0.05, 0.01),
(39, 'Lesotho', 5.85, 0.35, 0.45, 0.0, 0.05, 0.05, 0.01),
(40, 'Swaziland', 5.8, 0.3, 0.4, 0.0, 0.05, 0.05, 0.01),
(41, 'Mali', 5.75, 0.25, 0.35, 0.0, 0.05, 0.05, 0.01),
(42, 'Niger', 5.7, 0.2, 0.3, 0.0, 0.05, 0.05, 0.01),
(43, 'Chad', 5.65, 0.15, 0.25, 0.0, 0.05, 0.05, 0.01),
(44, 'Sudan', 5.6, 0.1, 0.2, 0.0, 0.05, 0.05, 0.01),
(45, 'South Sudan', 5.55, 0.05, 0.15, 0.0, 0.05, 0.05, 0.01),
(46, 'Congo', 5.5, 0.0, 0.1, 0.0, 0.05, 0.05, 0.01),
(47, 'Cote d'Ivoire', 5.45, 0.0, 0.05, 0.0, 0.05, 0.05, 0.01),
(48, 'Ghana', 5.4, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(49, 'Sierra Leone', 5.35, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(50, 'Liberia', 5.3, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(51, 'Ivory Coast', 5.25, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(52, 'Senegal', 5.2, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(53, 'Gambia', 5.15, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(54, 'Guinea', 5.1, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(55, 'Sierra Leone', 5.05, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(56, 'Liberia', 5.0, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(57, 'Ivory Coast', 4.95, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(58, 'Senegal', 4.9, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(59, 'Gambia', 4.85, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(60, 'Guinea', 4.8, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(61, 'Sierra Leone', 4.75, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(62, 'Liberia', 4.7, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(63, 'Ivory Coast', 4.65, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(64, 'Senegal', 4.6, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(65, 'Gambia', 4.55, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(66, 'Guinea', 4.5, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(67, 'Sierra Leone', 4.45, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(68, 'Liberia', 4.4, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(69, 'Ivory Coast', 4.35, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(70, 'Senegal', 4.3, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(71, 'Gambia', 4.25, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(72, 'Guinea', 4.2, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(73, 'Sierra Leone', 4.15, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(74, 'Liberia', 4.1, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(75, 'Ivory Coast', 4.05, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(76, 'Senegal', 4.0, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(77, 'Gambia', 3.95, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(78, 'Guinea', 3.9, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(79, 'Sierra Leone', 3.85, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(80, 'Liberia', 3.8, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(81, 'Ivory Coast', 3.75, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(82, 'Senegal', 3.7, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(83, 'Gambia', 3.65, 0.0, 0.0, 0.0, 0.05, 0.05, 0.01),
(84, 'Guinea
```

```
In [384]: webdf3.to_sql('webfinal1',conn)
```

```
C:\Users\nneam\anaconda3\lib\site-packages\pandas\core\generic.py:2653: UserWarning: The spaces in these column names
will not be changed. In pandas versions < 0.14, spaces were converted to underscores.
    sql.to_sql(
```

```
In [385]: #Executing Query  
c.execute("SELECT * FROM webfinal1").fetchall()
```

```
Out[385]: [(0, 'China', '1,388,232,693', '37', 0.185),  
(1, 'India', '1,342,512,706', '27', 0.179),  
(2, 'United States', '326,474,013', '38', 0.043),  
(3, 'Indonesia', '263,510,146', '28', 0.035),  
(4, 'Brazil', '211,243,220', '31', 0.027999999999999997),  
(5, 'Pakistan', '196,744,376', '23', 0.026000000000000002),  
(6, 'Nigeria', '191,835,936', '18', 0.026000000000000002),  
(7, 'Bangladesh', '164,827,718', '26', 0.022000000000000002),  
(8, 'Russia', '143,375,006', '39', 0.019),  
(9, 'Mexico', '130,222,815', '27', 0.017),  
(10, 'Japan', '126,045,211', '47', 0.017),  
(11, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(12, 'Philippines', '103,796,832', '24', 0.013999999999999999),  
(13, 'Viet Nam', '95,414,640', '30', 0.013000000000000001),  
(14, 'Egypt', '95,215,102', '25', 0.013000000000000001),  
(15, 'DR Congo', '82,242,685', '17', 0.011000000000000001),  
(16, 'Iran', '80,945,718', '30', 0.011000000000000001),  
(17, 'Germany', '80,636,124', '46', 0.011000000000000001),  
(18, 'Turkey', '80,417,526', '30', 0.011000000000000001),  
(19, 'Thailand', '66,227,547', '18', 0.009999999999999999),  
(20, 'France', '66,227,547', '48', 0.009999999999999999),  
(21, 'Canada', '38,232,123', '55', 0.009999999999999999),  
(22, 'Australia', '25,495,476', '35', 0.009999999999999999),  
(23, 'South Korea', '51,709,323', '36', 0.009999999999999999),  
(24, 'Italy', '60,725,204', '44', 0.009999999999999999),  
(25, 'Spain', '46,754,779', '41', 0.009999999999999999),  
(26, 'Netherlands', '17,172,300', '52', 0.009999999999999999),  
(27, 'Sweden', '10,129,200', '59', 0.009999999999999999),  
(28, 'Belgium', '11,589,621', '53', 0.009999999999999999),  
(29, 'Austria', '9,006,400', '49', 0.009999999999999999),  
(30, 'Switzerland', '8,501,132', '45', 0.009999999999999999),  
(31, 'Denmark', '5,524,583', '56', 0.009999999999999999),  
(32, 'Norway', '5,370,326', '58', 0.009999999999999999),  
(33, 'Finland', '5,541,320', '64', 0.009999999999999999),  
(34, 'Ireland', '4,548,266', '54', 0.009999999999999999),  
(35, 'Portugal', '10,632,666', '39', 0.009999999999999999),  
(36, 'Greece', '11,516,654', '38', 0.009999999999999999),  
(37, 'Czech Republic', '10,506,800', '49', 0.009999999999999999),  
(38, 'Poland', '38,438,796', '51', 0.009999999999999999),  
(39, 'Hungary', '10,381,151', '47', 0.009999999999999999),  
(40, 'Slovakia', '5,459,692', '48', 0.009999999999999999),  
(41, 'Slovenia', '2,064,196', '46', 0.009999999999999999),  
(42, 'Croatia', '4,287,593', '45', 0.009999999999999999),  
(43, 'Serbia', '7,123,271', '44', 0.009999999999999999),  
(44, 'Bosnia and Herzegovina', '3,531,292', '45', 0.009999999999999999),  
(45, 'Montenegro', '628,000', '43', 0.009999999999999999),  
(46, 'Albania', '2,876,329', '39', 0.009999999999999999),  
(47, 'Moldova', '4,130,830', '48', 0.009999999999999999),  
(48, 'Ukraine', '47,792,349', '49', 0.009999999999999999),  
(49, 'Belarus', '9,596,464', '53', 0.009999999999999999),  
(50, 'Lithuania', '3,302,283', '56', 0.009999999999999999),  
(51, 'Latvia', '1,360,696', '58', 0.009999999999999999),  
(52, 'Estonia', '1,315,786', '59', 0.009999999999999999),  
(53, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(54, 'Malta', '441,540', '35', 0.009999999999999999),  
(55, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(56, 'Iceland', '341,227', '64', 0.009999999999999999),  
(57, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(58, 'San Marino', '33,973', '43', 0.009999999999999999),  
(59, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(60, 'Monaco', '34,000', '43', 0.009999999999999999),  
(61, 'Andorra', '78,000', '42', 0.009999999999999999),  
(62, 'Nauru', '11,824', '1', 0.009999999999999999),  
(63, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(64, 'Palau', '16,600', '7', 0.009999999999999999),  
(65, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(66, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(67, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(68, 'Bhutan', '752,000', '27', 0.009999999999999999),  
(69, 'Maldives', '344,000', '3', 0.009999999999999999),  
(70, 'Yemen', '28,320,000', '15', 0.009999999999999999),  
(71, 'Somalia', '15,853,000', '1', 0.009999999999999999),  
(72, 'Djibouti', '992,000', '11', 0.009999999999999999),  
(73, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(74, 'Kenya', '44,299,000', '1', 0.009999999999999999),  
(75, 'Tanzania', '55,155,000', '4', 0.009999999999999999),  
(76, 'Uganda', '34,600,000', '1', 0.009999999999999999),  
(77, 'Rwanda', '12,130,000', '1', 0.009999999999999999),  
(78, 'Burundi', '10,100,000', '1', 0.009999999999999999),  
(79, 'Mali', '20,254,000', '1', 0.009999999999999999),  
(80, 'Niger', '24,207,000', '1', 0.009999999999999999),  
(81, 'Chad', '16,736,000', '1', 0.009999999999999999),  
(82, 'Sudan', '43,840,000', '1', 0.009999999999999999),  
(83, 'South Sudan', '11,208,000', '1', 0.009999999999999999),  
(84, 'Eritrea', '5,342,000', '1', 0.009999999999999999),  
(85, 'Libya', '6,802,000', '1', 0.009999999999999999),  
(86, 'Syria', '22,645,000', '1', 0.009999999999999999),  
(87, 'Iraq', '38,327,000', '1', 0.009999999999999999),  
(88, 'Jordan', '9,596,000', '1', 0.009999999999999999),  
(89, 'Lebanon', '6,013,000', '1', 0.009999999999999999),  
(90, 'Israel', '8,526,000', '1', 0.009999999999999999),  
(91, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(92, 'Malta', '441,540', '35', 0.009999999999999999),  
(93, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(94, 'Iceland', '341,227', '64', 0.009999999999999999),  
(95, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(96, 'San Marino', '33,973', '43', 0.009999999999999999),  
(97, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(98, 'Monaco', '34,000', '43', 0.009999999999999999),  
(99, 'Andorra', '78,000', '42', 0.009999999999999999),  
(100, 'Nauru', '11,824', '1', 0.009999999999999999),  
(101, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(102, 'Palau', '16,600', '7', 0.009999999999999999),  
(103, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(104, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(105, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(106, 'Bhutan', '752,000', '27', 0.009999999999999999),  
(107, 'Maldives', '344,000', '3', 0.009999999999999999),  
(108, 'Yemen', '28,320,000', '15', 0.009999999999999999),  
(109, 'Somalia', '15,853,000', '1', 0.009999999999999999),  
(110, 'Djibouti', '992,000', '11', 0.009999999999999999),  
(111, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(112, 'Kenya', '44,299,000', '1', 0.009999999999999999),  
(113, 'Tanzania', '55,155,000', '4', 0.009999999999999999),  
(114, 'Uganda', '34,600,000', '1', 0.009999999999999999),  
(115, 'Rwanda', '12,130,000', '1', 0.009999999999999999),  
(116, 'Burundi', '10,100,000', '1', 0.009999999999999999),  
(117, 'Mali', '20,254,000', '1', 0.009999999999999999),  
(118, 'Niger', '24,207,000', '1', 0.009999999999999999),  
(119, 'Chad', '16,736,000', '1', 0.009999999999999999),  
(120, 'Sudan', '43,840,000', '1', 0.009999999999999999),  
(121, 'South Sudan', '11,208,000', '1', 0.009999999999999999),  
(122, 'Eritrea', '5,342,000', '1', 0.009999999999999999),  
(123, 'Libya', '6,802,000', '1', 0.009999999999999999),  
(124, 'Syria', '22,645,000', '1', 0.009999999999999999),  
(125, 'Iraq', '38,327,000', '1', 0.009999999999999999),  
(126, 'Jordan', '9,596,000', '1', 0.009999999999999999),  
(127, 'Lebanon', '6,013,000', '1', 0.009999999999999999),  
(128, 'Israel', '8,526,000', '1', 0.009999999999999999),  
(129, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(130, 'Malta', '441,540', '35', 0.009999999999999999),  
(131, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(132, 'Iceland', '341,227', '64', 0.009999999999999999),  
(133, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(134, 'San Marino', '33,973', '43', 0.009999999999999999),  
(135, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(136, 'Monaco', '34,000', '43', 0.009999999999999999),  
(137, 'Andorra', '78,000', '42', 0.009999999999999999),  
(138, 'Nauru', '11,824', '1', 0.009999999999999999),  
(139, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(140, 'Palau', '16,600', '7', 0.009999999999999999),  
(141, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(142, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(143, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(144, 'Bhutan', '752,000', '27', 0.009999999999999999),  
(145, 'Maldives', '344,000', '3', 0.009999999999999999),  
(146, 'Yemen', '28,320,000', '15', 0.009999999999999999),  
(147, 'Somalia', '15,853,000', '1', 0.009999999999999999),  
(148, 'Djibouti', '992,000', '11', 0.009999999999999999),  
(149, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(150, 'Kenya', '44,299,000', '1', 0.009999999999999999),  
(151, 'Tanzania', '55,155,000', '4', 0.009999999999999999),  
(152, 'Uganda', '34,600,000', '1', 0.009999999999999999),  
(153, 'Rwanda', '12,130,000', '1', 0.009999999999999999),  
(154, 'Burundi', '10,100,000', '1', 0.009999999999999999),  
(155, 'Mali', '20,254,000', '1', 0.009999999999999999),  
(156, 'Niger', '24,207,000', '1', 0.009999999999999999),  
(157, 'Chad', '16,736,000', '1', 0.009999999999999999),  
(158, 'Sudan', '43,840,000', '1', 0.009999999999999999),  
(159, 'South Sudan', '11,208,000', '1', 0.009999999999999999),  
(160, 'Eritrea', '5,342,000', '1', 0.009999999999999999),  
(161, 'Libya', '6,802,000', '1', 0.009999999999999999),  
(162, 'Syria', '22,645,000', '1', 0.009999999999999999),  
(163, 'Iraq', '38,327,000', '1', 0.009999999999999999),  
(164, 'Jordan', '9,596,000', '1', 0.009999999999999999),  
(165, 'Lebanon', '6,013,000', '1', 0.009999999999999999),  
(166, 'Israel', '8,526,000', '1', 0.009999999999999999),  
(167, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(168, 'Malta', '441,540', '35', 0.009999999999999999),  
(169, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(170, 'Iceland', '341,227', '64', 0.009999999999999999),  
(171, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(172, 'San Marino', '33,973', '43', 0.009999999999999999),  
(173, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(174, 'Monaco', '34,000', '43', 0.009999999999999999),  
(175, 'Andorra', '78,000', '42', 0.009999999999999999),  
(176, 'Nauru', '11,824', '1', 0.009999999999999999),  
(177, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(178, 'Palau', '16,600', '7', 0.009999999999999999),  
(179, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(180, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(181, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(182, 'Bhutan', '752,000', '27', 0.009999999999999999),  
(183, 'Maldives', '344,000', '3', 0.009999999999999999),  
(184, 'Yemen', '28,320,000', '15', 0.009999999999999999),  
(185, 'Somalia', '15,853,000', '1', 0.009999999999999999),  
(186, 'Djibouti', '992,000', '11', 0.009999999999999999),  
(187, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(188, 'Kenya', '44,299,000', '1', 0.009999999999999999),  
(189, 'Tanzania', '55,155,000', '4', 0.009999999999999999),  
(190, 'Uganda', '34,600,000', '1', 0.009999999999999999),  
(191, 'Rwanda', '12,130,000', '1', 0.009999999999999999),  
(192, 'Burundi', '10,100,000', '1', 0.009999999999999999),  
(193, 'Mali', '20,254,000', '1', 0.009999999999999999),  
(194, 'Niger', '24,207,000', '1', 0.009999999999999999),  
(195, 'Chad', '16,736,000', '1', 0.009999999999999999),  
(196, 'Sudan', '43,840,000', '1', 0.009999999999999999),  
(197, 'South Sudan', '11,208,000', '1', 0.009999999999999999),  
(198, 'Eritrea', '5,342,000', '1', 0.009999999999999999),  
(199, 'Libya', '6,802,000', '1', 0.009999999999999999),  
(200, 'Syria', '22,645,000', '1', 0.009999999999999999),  
(201, 'Iraq', '38,327,000', '1', 0.009999999999999999),  
(202, 'Jordan', '9,596,000', '1', 0.009999999999999999),  
(203, 'Lebanon', '6,013,000', '1', 0.009999999999999999),  
(204, 'Israel', '8,526,000', '1', 0.009999999999999999),  
(205, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(206, 'Malta', '441,540', '35', 0.009999999999999999),  
(207, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(208, 'Iceland', '341,227', '64', 0.009999999999999999),  
(209, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(210, 'San Marino', '33,973', '43', 0.009999999999999999),  
(211, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(212, 'Monaco', '34,000', '43', 0.009999999999999999),  
(213, 'Andorra', '78,000', '42', 0.009999999999999999),  
(214, 'Nauru', '11,824', '1', 0.009999999999999999),  
(215, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(216, 'Palau', '16,600', '7', 0.009999999999999999),  
(217, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(218, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(219, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(220, 'Bhutan', '752,000', '27', 0.009999999999999999),  
(221, 'Maldives', '344,000', '3', 0.009999999999999999),  
(222, 'Yemen', '28,320,000', '15', 0.009999999999999999),  
(223, 'Somalia', '15,853,000', '1', 0.009999999999999999),  
(224, 'Djibouti', '992,000', '11', 0.009999999999999999),  
(225, 'Ethiopia', '104,344,901', '19', 0.013999999999999999),  
(226, 'Kenya', '44,299,000', '1', 0.009999999999999999),  
(227, 'Tanzania', '55,155,000', '4', 0.009999999999999999),  
(228, 'Uganda', '34,600,000', '1', 0.009999999999999999),  
(229, 'Rwanda', '12,130,000', '1', 0.009999999999999999),  
(230, 'Burundi', '10,100,000', '1', 0.009999999999999999),  
(231, 'Mali', '20,254,000', '1', 0.009999999999999999),  
(232, 'Niger', '24,207,000', '1', 0.009999999999999999),  
(233, 'Chad', '16,736,000', '1', 0.009999999999999999),  
(234, 'Sudan', '43,840,000', '1', 0.009999999999999999),  
(235, 'South Sudan', '11,208,000', '1', 0.009999999999999999),  
(236, 'Eritrea', '5,342,000', '1', 0.009999999999999999),  
(237, 'Libya', '6,802,000', '1', 0.009999999999999999),  
(238, 'Syria', '22,645,000', '1', 0.009999999999999999),  
(239, 'Iraq', '38,327,000', '1', 0.009999999999999999),  
(240, 'Jordan', '9,596,000', '1', 0.009999999999999999),  
(241, 'Lebanon', '6,013,000', '1', 0.009999999999999999),  
(242, 'Israel', '8,526,000', '1', 0.009999999999999999),  
(243, 'Cyprus', '848,864', '35', 0.009999999999999999),  
(244, 'Malta', '441,540', '35', 0.009999999999999999),  
(245, 'Luxembourg', '583,991', '47', 0.009999999999999999),  
(246, 'Iceland', '341,227', '64', 0.009999999999999999),  
(247, 'Liechtenstein', '37,631', '48', 0.009999999999999999),  
(248, 'San Marino', '33,973', '43', 0.009999999999999999),  
(249, 'Vatican City', '100,000', '43', 0.009999999999999999),  
(250, 'Monaco', '34,000', '43', 0.009999999999999999),  
(251, 'Andorra', '78,000', '42', 0.009999999999999999),  
(252, 'Nauru', '11,824', '1', 0.009999999999999999),  
(253, 'Tuvalu', '11,780', '7', 0.009999999999999999),  
(254, 'Palau', '16,600', '7', 0.009999999999999999),  
(255, 'Marshall Islands', '59,192', '9', 0.009999999999999999),  
(256, 'Micronesia', '111,935', '7', 0.009999999999999999),  
(257, 'Nepal', '28,320,000', '28', 0.009999999999999999),  
(258, 'Bhutan', '752,000', '27',
```

```
In [378]: #Executing Query  
c.execute("SELECT * FROM apifinal1").fetchall()
```

```
Out[378]: [(141, 'Isle of Man', None, 39863.0, 44214.0, None, None, None, None),  
(142,  
  'Israel',  
  29.4460992004691,  
  673494.0,  
  8209306.0,  
  4102822.0,  
  3.0,  
  82.8024390243903,  
  46.18838654478318),  
(143,  
  'Italy',  
  31.4517439505479,  
  17861881.0,  
  42559879.0,  
  26034264.0,  
  2.8,  
  83.3463414634146,  
  43.08756315605503),  
  ...]
```

```
In [379]: # conn.commit()
```

```
In [399]: #Assigning SQL table to df  
  
apidata = pd.read_sql_query('SELECT * FROM apifinal1', conn)
```

```
In [400]: #Assigning SQL table to df  
  
webdata = pd.read_sql_query('SELECT * FROM webfinal1', conn)
```

```
In [401]: #Assigning SQL table to df  
csvdata = pd.read_sql_query('SELECT * FROM csvf1', conn)
```

```
In [433]: #Merging all datasets together to meet requirements  
final = pd.merge(pd.merge(apidata,webdata,on='Country'),csvdata,on='Country')
```

In [434]: `final.head(300)`

Out[434]:

	index_x	Country	GDP_%_of_Exports	Rural_Pop	Urban_Pop	Available_Workers	Infant_Mortality_per_1000	Life_Expectancy	%_of_i
0	142	Israel	29.446099	673494.0	8209306.0	4102822.0	3.0	82.802439	
1	143	Italy	31.451744	17861881.0	42559879.0	26034264.0	2.8	83.346341	
2	144	Jamaica	37.957853	1300904.0	1633951.0	1473383.0	12.3	74.368000	
3	145	Japan	18.524909	10608200.0	115920900.0	68358370.0	1.8	84.210976	
4	146	Jordan	35.638746	898132.0	9057879.0	2579658.0	13.8	74.405000	
5	147	Kazakhstan	37.625226	7780670.5	10495828.0	9030838.0	9.2	73.150000	
6	148	Kenya	13.174140	37501479.0	13891531.0	23057935.0	32.8	66.342000	
7	151	South Korea	41.630856	9568386.0	42038247.0	28272711.0	2.8	82.626829	
8	153	Kuwait	56.720557	0.0	4137312.0	2383249.0	6.9	75.398000	
9	154	Kyrgyzstan	31.601730	4024399.0	2298401.0	2564912.0	17.1	71.400000	

In [436]: `#Final Clean up`  
`final = final.drop(columns=['index_y'])`  
`final = final.drop(columns=['index_x'])`  
`final = final.drop(columns=['index'])`  
`final = final.drop(columns=['Score'])`



```
In [440]: #Round Decimals and add , for seperation  
final.round({'GDP_%_of_Exports': 2, 'Infant_Mortality_per_1000': 1, 'Life_Expectancy': 1, '%_of_available_workers' :1})
```

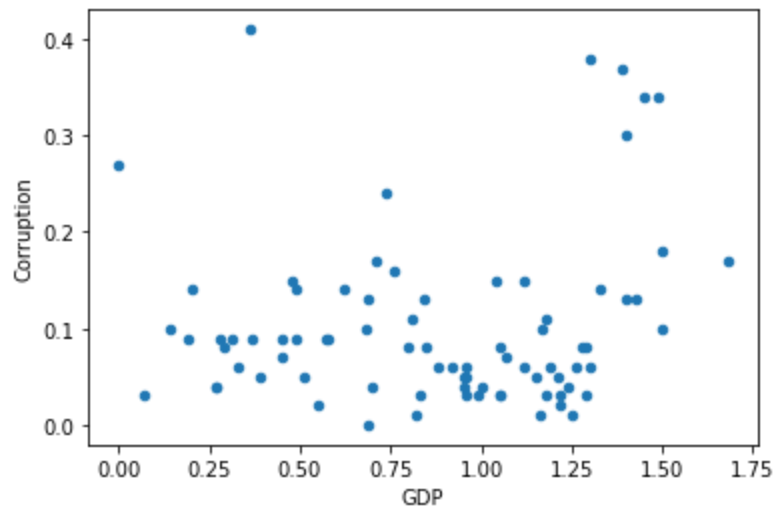
Out[440]:

	Country	GDP_%_of_Exports	Rural_Pop	Urban_Pop	Available_Workers	Infant_Mortality_per_1000	Life_Expectancy	%_of_available_
0	Israel	29.45	673494.0	8209306.0	4102822.0	3.0	82.8	
1	Italy	31.45	17861881.0	42559879.0	26034264.0	2.8	83.3	
2	Jamaica	37.96	1300904.0	1633951.0	1473383.0	12.3	74.4	
3	Japan	18.52	10608200.0	115920900.0	68358370.0	1.8	84.2	
4	Jordan	35.64	898132.0	9057879.0	2579658.0	13.8	74.4	
5	Kazakhstan	37.63	7780670.5	10495828.0	9030838.0	9.2	73.2	
6	Kenya	13.17	37501479.0	13891531.0	23057935.0	32.8	66.3	
7	South Korea	41.63	9568386.0	42038247.0	28272711.0	2.8	82.6	
8	Kuwait	56.72	0.0	4137312.0	2383249.0	6.9	75.4	
9	Kyrgyzstan	31.60	4024399.0	2298401.0	2564912.0	17.1	71.4	

## Visualizations

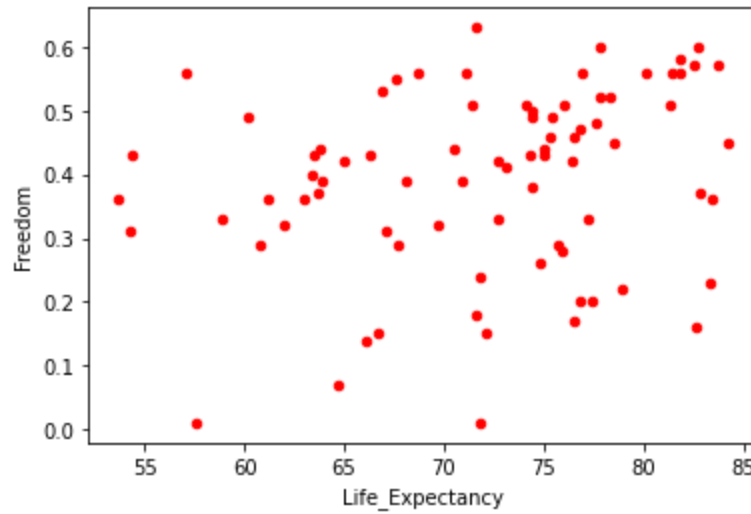
```
In [426]: #Analyzing trend of corruption over GDP  
final.plot.scatter(x = 'GDP', y = 'Corruption')
```

Out[426]: <matplotlib.axes.\_subplots.AxesSubplot at 0x25d0a8fa7c0>



```
In [428]: #Analyzing if freedom plays a role in life expectancy  
final.plot.scatter(x = 'Life_Expectancy', y = 'Freedom', c = 'Red')
```

```
Out[428]: <matplotlib.axes._subplots.AxesSubplot at 0x25d0ad0c4f0>
```



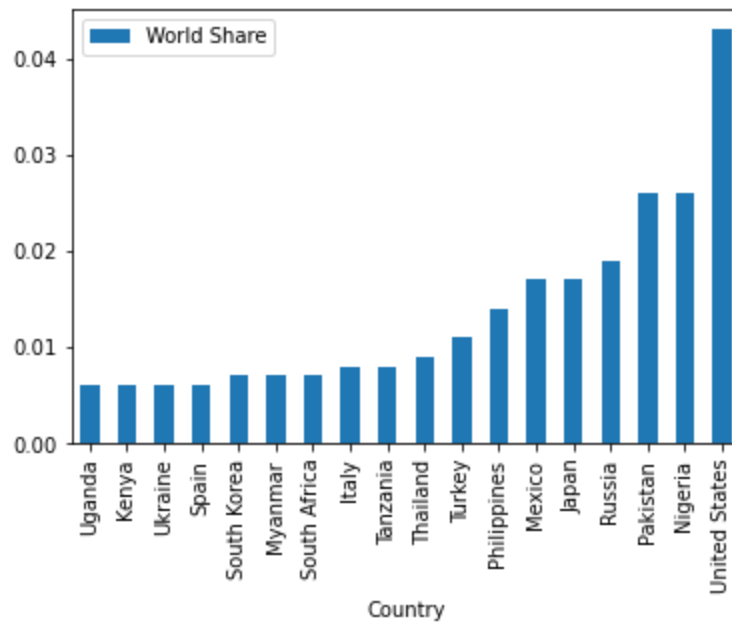
```
In [441]: #Finding mean of world share  
final['World Share'].mean()
```

```
Out[441]: 0.004337500000000001
```


```
In [443]: # Sorting values by world share and rounding up from mean  
countries = final[final['World Share'] > 0.005].sort_values('World Share')
```


```
In [446]: #Percent of world share  
countries.plot(x='Country', y = 'World Share', kind = 'bar')
```

```
Out[446]: <matplotlib.axes._subplots.AxesSubplot at 0x25d0b0fd250>
```



```
In [449]: import seaborn as sb
```

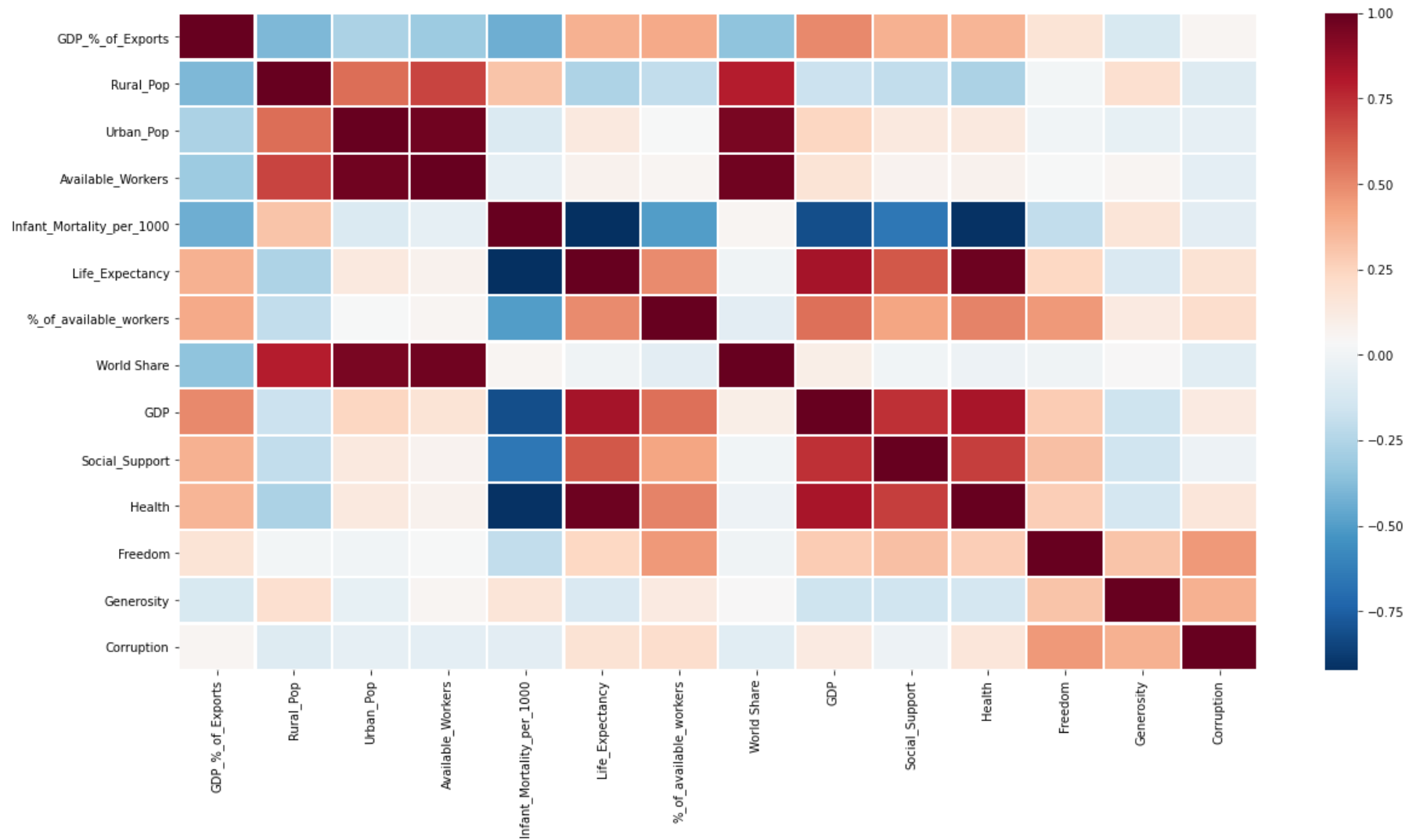
```
In [450]:  #Dropping Country Column for correlation matrix  
final_corr = final.drop(columns=['Country'])
```

```
In [452]:  #Preparing correlation  
pearson = final_corr.corr(method = 'pearson')
```

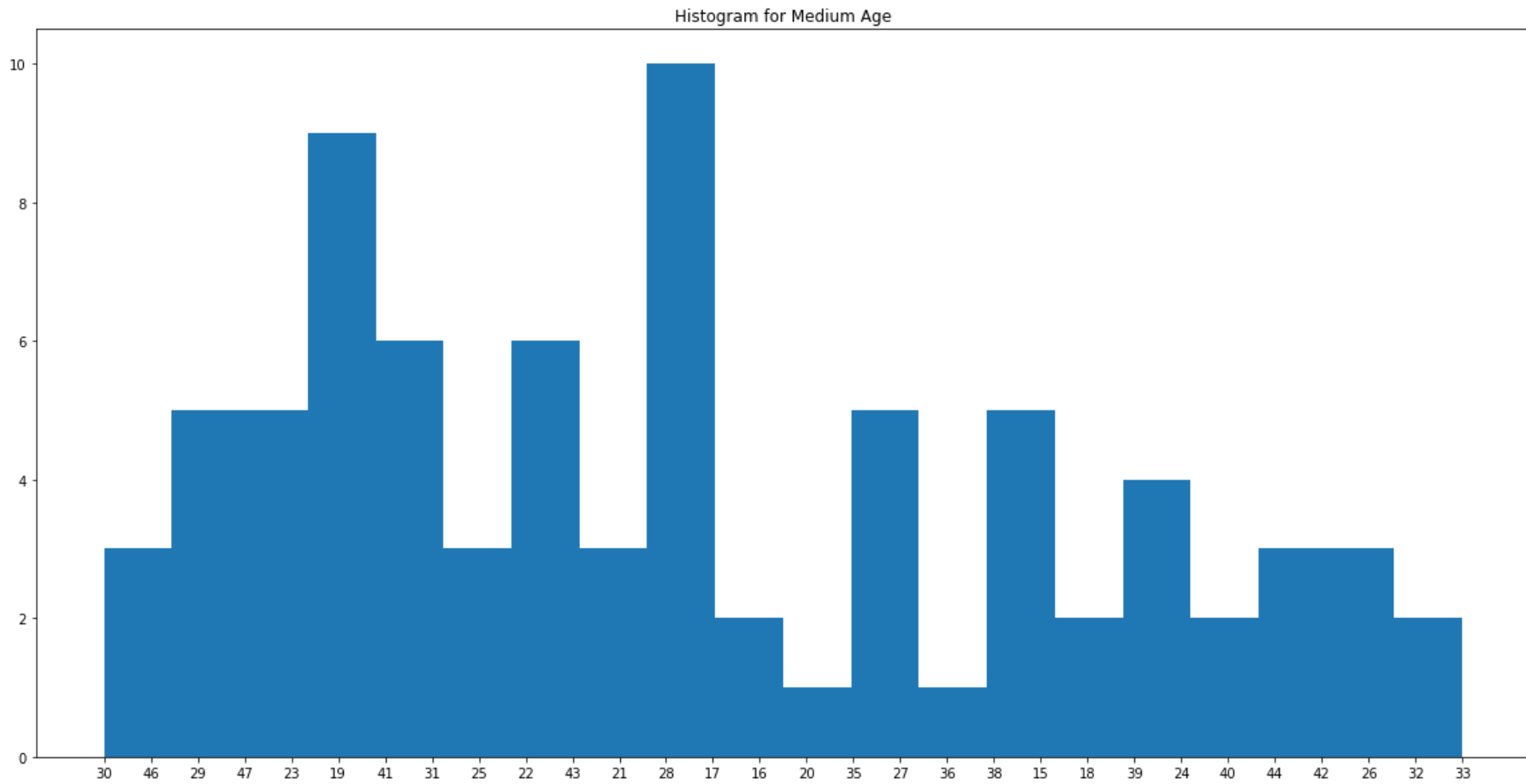
In [475]: `#Pearson Correlation to see what impacts eachother`

```
sb.heatmap(pearson,  
            xticklabels=pearson.columns,  
            yticklabels=pearson.columns,  
            cmap='RdBu_r',  
            annot=False,  
            linewidth=1.2)
```

Out[475]: `<matplotlib.axes._subplots.AxesSubplot at 0x25d0dbb3520>`

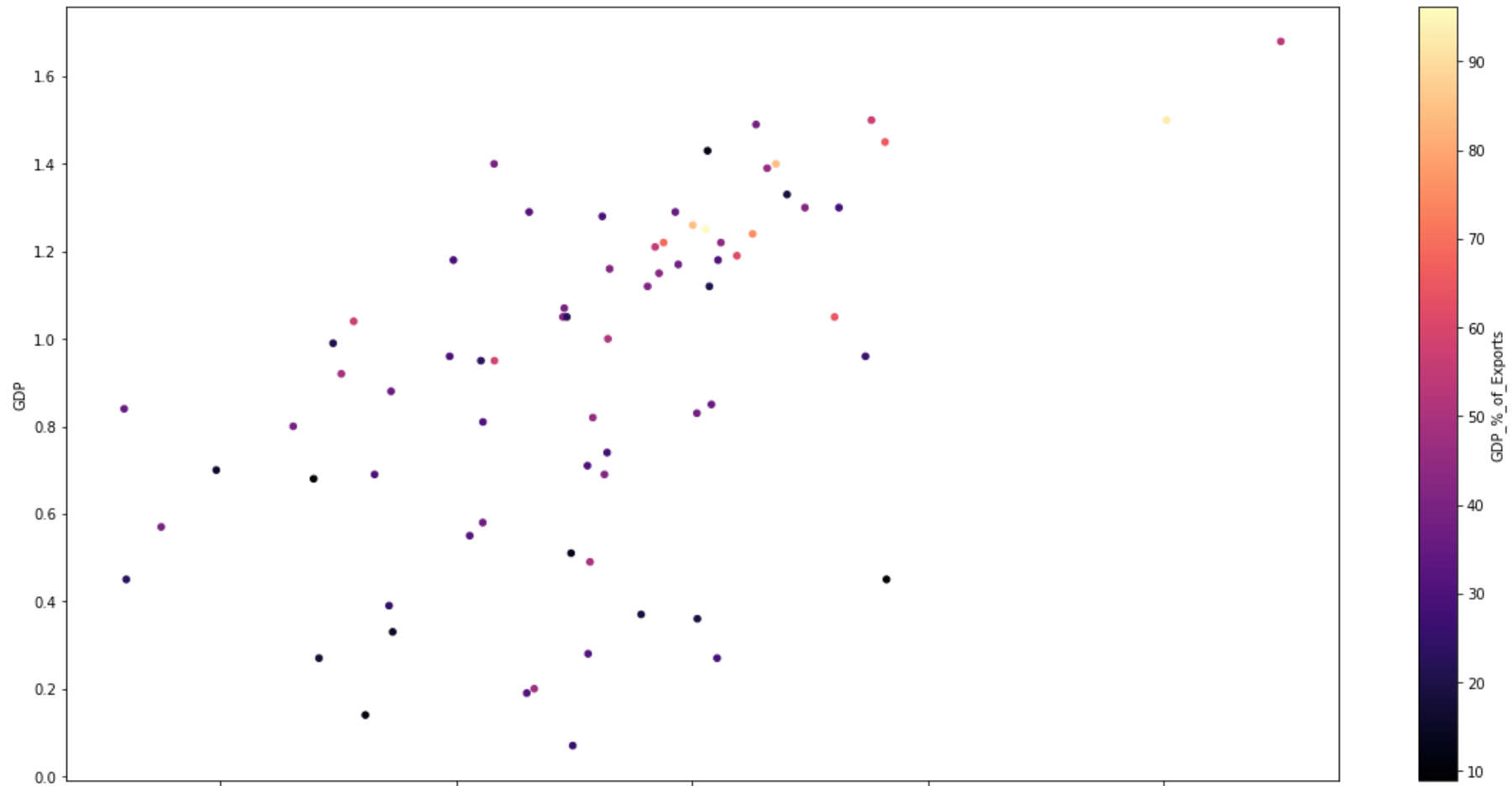


```
In [477]: #Histogram on Medium Age  
plt.title("Histogram for Medium Age")  
plt.hist(final['Med. Age'],bins=20)  
plt.show()
```



```
In [491]: #Checking scatter plot to analyze correlation between available workers and overall GDP.  
final.plot.scatter(x = '%_of_available_workers', y = 'GDP', c = 'GDP_%_of_Exports', colormap = 'magma')
```

```
Out[491]: <matplotlib.axes._subplots.AxesSubplot at 0x25d139087f0>
```



```
In [495]: conn.commit()
```

```
In [496]: conn.close()
```

