

Welcome to Covid19 Data Analysis Notebook

Let's Import the modules

```
In [8]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
print('Modules are imported.')
```

Modules are imported.

Task 2

Task 2.1: importing covid19 dataset

importing "Covid19_Confirmed_dataset.csv" from "./Dataset" folder.

```
In [9]: corona_dataset_csv = pd.read_csv("Datasets/covid19_Confirmed_dataset.csv")
corona_dataset_csv.head(10)
```

Out[9]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0
5	NaN	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0
6	NaN	Argentina	-38.4161	-63.6167	0	0	0	0	0
7	NaN	Armenia	40.0691	45.0382	0	0	0	0	0
8	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0
9	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3

10 rows × 104 columns



Let's check the shape of the dataframe

```
In [10]: corona_dataset_csv.shape
```

Out[10]: (266, 104)

Task 2.2: Delete the useless columns

```
In [11]: corona_dataset_csv.drop(["Lat", "Long"], axis=1, inplace=True)
```

In [16]: `corona_dataset_csv.head(10)`

Out[16]:

	Province/State	Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20
0	NaN	Afghanistan	0	0	0	0	0	0	0	0
1	NaN	Albania	0	0	0	0	0	0	0	0
2	NaN	Algeria	0	0	0	0	0	0	0	0
3	NaN	Andorra	0	0	0	0	0	0	0	0
4	NaN	Angola	0	0	0	0	0	0	0	0
5	NaN	Antigua and Barbuda	0	0	0	0	0	0	0	0
6	NaN	Argentina	0	0	0	0	0	0	0	0
7	NaN	Armenia	0	0	0	0	0	0	0	0
8	Australian Capital Territory	Australia	0	0	0	0	0	0	0	0
9	New South Wales	Australia	0	0	0	0	3	4	4	4

10 rows × 102 columns



Task 2.3: Aggregating the rows by the country

In [12]: `corona_dataset_aggregated = corona_dataset_csv.groupby("Country/Region").sum()`

In [13]: `corona_dataset_aggregated.head()`

Out[13]:

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20
Country/Region										
Afghanistan	0	0	0	0	0	0	0	0	0	0
Albania	0	0	0	0	0	0	0	0	0	0
Algeria	0	0	0	0	0	0	0	0	0	0
Andorra	0	0	0	0	0	0	0	0	0	0
Angola	0	0	0	0	0	0	0	0	0	0

5 rows × 100 columns



In [14]: `corona_dataset_aggregated.shape`

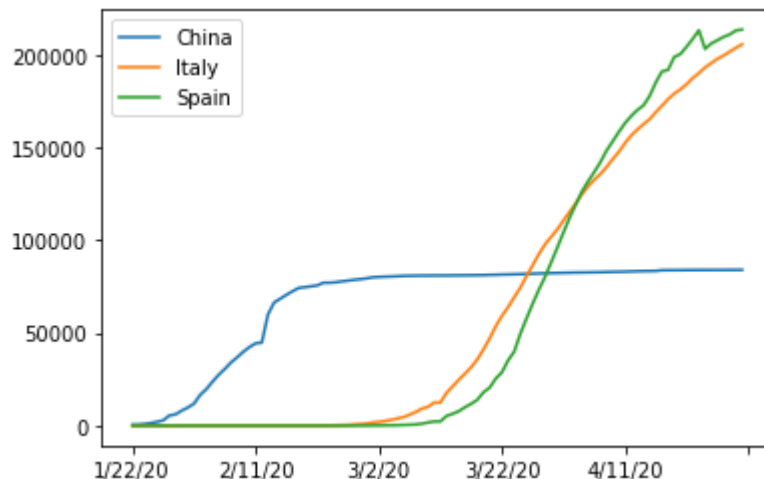
Out[14]: (187, 100)

Task 2.4: Visualizing data related to a country for example China

visualization always helps for better understanding of our data.

```
In [15]: corona_dataset_aggregated.loc["China"].plot()  
corona_dataset_aggregated.loc["Italy"].plot()  
corona_dataset_aggregated.loc["Spain"].plot()  
plt.legend()
```

Out[15]: <matplotlib.legend.Legend at 0xccc51d8>

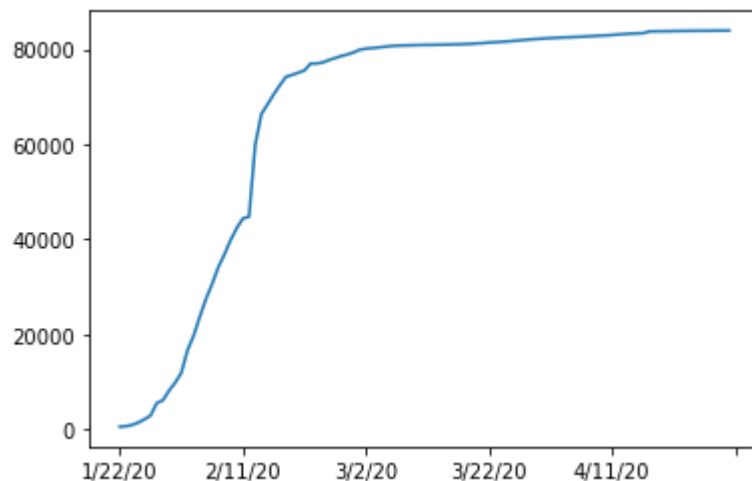


Task3: Calculating a good measure

we need to find a good measure represented as a number, describing the spread of the virus in a country.

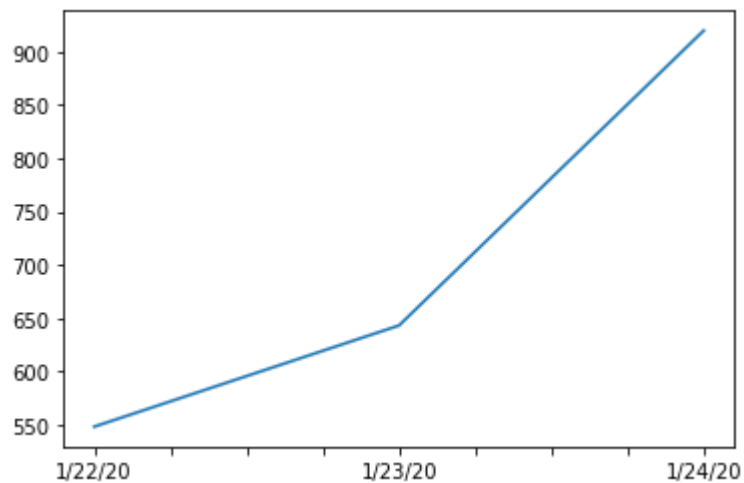
```
In [16]: corona_dataset_aggregated.loc['China'].plot()
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0xdd88178>



```
In [17]: corona_dataset_aggregated.loc['China'][:3].plot()
```

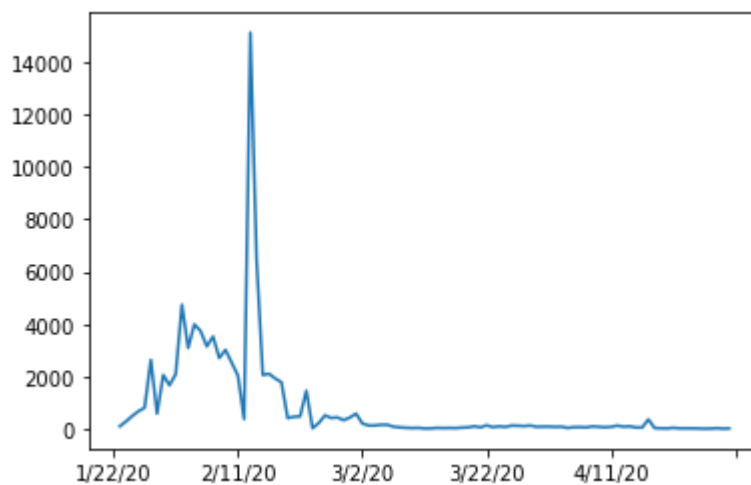
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0xde08d78>
```



task 3.1: caculating the first derivative of the curve

```
In [18]: corona_dataset_aggregated.loc['China'].diff().plot()
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x12b4a30>
```



task 3.2: find maximum infection rate for China

```
In [19]: corona_dataset_aggregated.loc['China'].diff().max()
```

```
Out[19]: 15136.0
```

```
In [20]: corona_dataset_aggregated.loc['Italy'].diff().max()
```

```
Out[20]: 6557.0
```

```
In [21]: corona_dataset_aggregated.loc['Spain'].diff().max()
```

```
Out[21]: 9630.0
```

Task 3.3: find maximum infection rate for all of the countries.

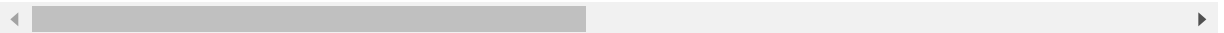
```
In [25]: countries = list(corona_dataset_aggregated.index)
max_infection_rates = []
for c in countries:
    max_infection_rates.append(corona_dataset_aggregated.loc[c].diff().max())
corona_dataset_aggregated["max_infection_rates"] = max_infection_rates
```

```
In [26]: corona_dataset_aggregated.head()
```

```
Out[26]:
```

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20
Country/Region										
Afghanistan	0	0	0	0	0	0	0	0	0	0
Albania	0	0	0	0	0	0	0	0	0	0
Algeria	0	0	0	0	0	0	0	0	0	0
Andorra	0	0	0	0	0	0	0	0	0	0
Angola	0	0	0	0	0	0	0	0	0	0

5 rows × 101 columns



Task 3.4: create a new dataframe with only needed column

```
In [36]: corona_data = pd.DataFrame(corona_dataset_aggregated["max_infection_rates"])
```

```
In [37]: corona_data.head()
```

```
Out[37]:
```

	max_infection_rates
Country/Region	
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

Task4:

- Importing the WorldHappinessReport.csv dataset
- selecting needed columns for our analysis
- join the datasets
- calculate the correlations as the result of our analysis

Task 4.1 : importing the dataset

```
In [39]: happiness_report_csv = pd.read_csv("Datasets/worldwide_happiness_report.csv")
```

```
In [40]: happiness_report_csv.head()
```

Out[40]:

	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

Task 4.2: let's drop the useless columns

```
In [41]: useless_cols = ["Overall rank", "Score", "Generosity", "Perceptions of corruption"]
```

```
In [42]: happiness_report_csv.drop(useless_cols, axis=1, inplace=True)
happiness_report_csv.head()
```

Out[42]:

	Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	1.340	1.587	0.986	0.596
1	Denmark	1.383	1.573	0.996	0.592
2	Norway	1.488	1.582	1.028	0.603
3	Iceland	1.380	1.624	1.026	0.591
4	Netherlands	1.396	1.522	0.999	0.557

Task 4.3: changing the indices of the dataframe

```
In [50]: happiness_report_csv.set_index("Country or region", inplace=True)
```

```
-----
KeyError                                Traceback (most recent call last)
<ipython-input-50-b6b651507760> in <module>
----> 1 happiness_report_csv.set_index("Country or region", inplace=True)

c:\users\administrator\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\frame.py in set_index(self, keys, drop, append, inplace, verify_integrity)
    4301
    4302         if missing:
-> 4303             raise KeyError(f"None of {missing} are in the columns")
    4304
    4305         if inplace:
```

KeyError: "None of ['Country or region'] are in the columns"

```
In [49]: happiness_report_csv.head()
```

Out[49]:

	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Country or region				
Finland	1.340	1.587	0.986	0.596
Denmark	1.383	1.573	0.996	0.592
Norway	1.488	1.582	1.028	0.603
Iceland	1.380	1.624	1.026	0.591
Netherlands	1.396	1.522	0.999	0.557

Task4.4: now let's join two dataset we have prepared

Corona Dataset :

In [51]: `corona_data.head()`

Out[51]:

	max_infection_rates
Country/Region	
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

In [53]: `corona_data.shape`

Out[53]: (187, 1)

world happiness report Dataset :

In [54]: `happiness_report_csv.head()`

Out[54]:

	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Country or region				
Finland	1.340	1.587	0.986	0.596
Denmark	1.383	1.573	0.996	0.592
Norway	1.488	1.582	1.028	0.603
Iceland	1.380	1.624	1.026	0.591
Netherlands	1.396	1.522	0.999	0.557

In [55]: `happiness_report_csv.shape`

Out[55]: (156, 4)

```
In [56]: data = corona_data.join(happiness_report_csv, how="inner")
data.head()
```

Out[56]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

Task 4.5: correlation matrix

```
In [57]: data.corr()
```

Out[57]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
max_infection_rates	1.000000	0.250118	0.191958	0.289263	0.078196
GDP per capita	0.250118	1.000000	0.759468	0.863062	0.394603
Social support	0.191958	0.759468	1.000000	0.765286	0.456246
Healthy life expectancy	0.289263	0.863062	0.765286	1.000000	0.427892
Freedom to make life choices	0.078196	0.394603	0.456246	0.427892	1.000000

Task 5: Visualization of the results

our Analysis is not finished unless we visualize the results in terms figures and graphs so that everyone can understand what you get out of our analysis

```
In [58]: data.head()
```

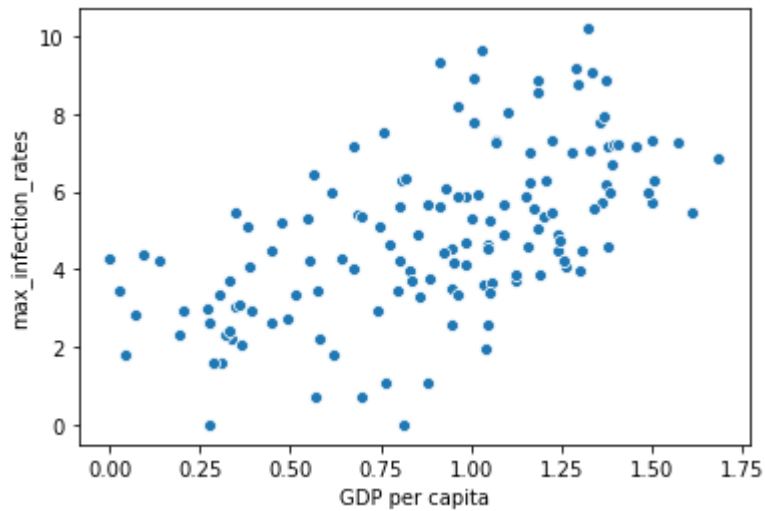
Out[58]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

Task 5.1: Plotting GDP vs maximum Infection rate

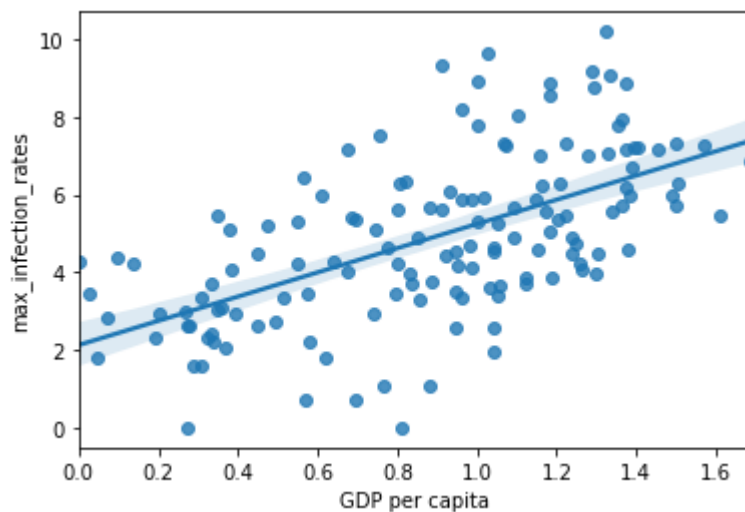
```
In [60]: x = data["GDP per capita"]  
y = data["max_infection_rates"]  
sns.scatterplot(x,np.log(y))
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0xe35b4f0>



```
In [61]: sns.regplot(x, np.log(y))
```

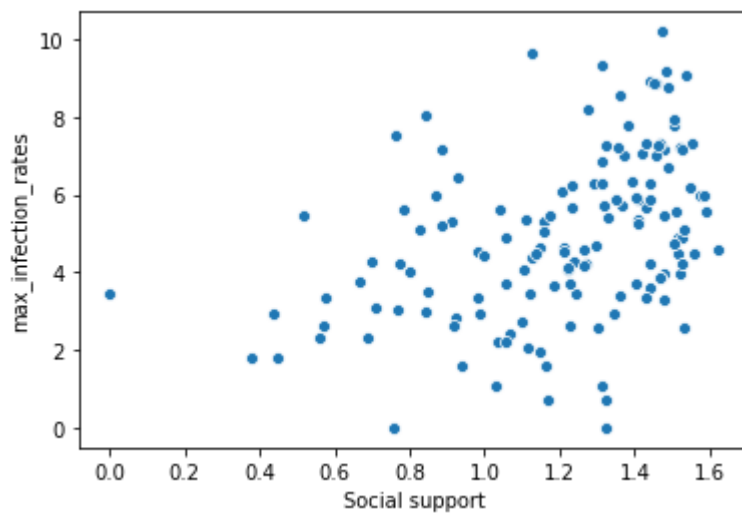
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0xe3812e0>



Task 5.2: Plotting Social support vs maximum Infection rate

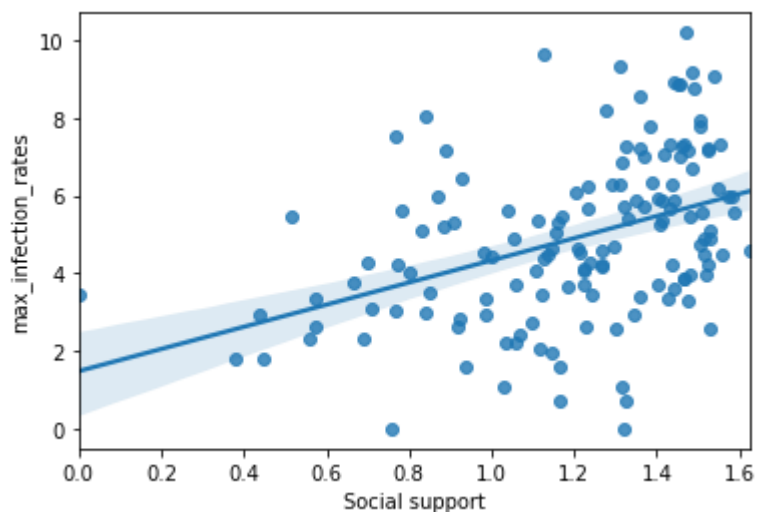
```
In [62]: x = data["Social support"]  
y = data["max_infection_rates"]  
sns.scatterplot(x,np.log(y))
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0xe3c9fa0>



```
In [63]: sns.regplot(x, np.log(y))
```

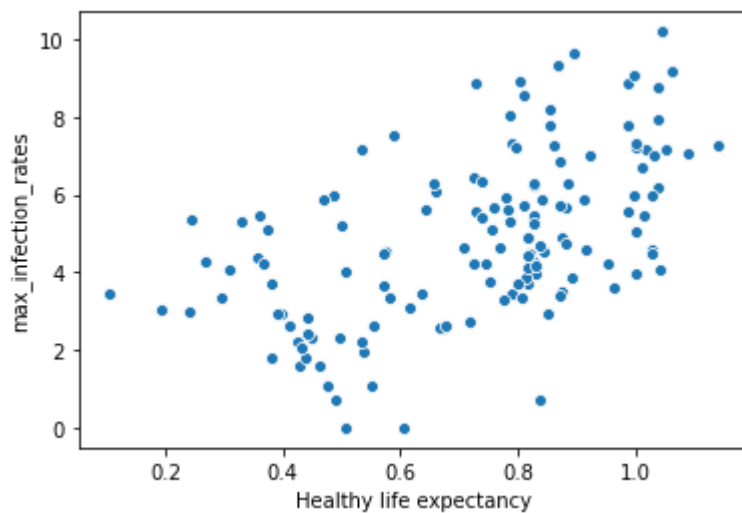
Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0xe3ffa00>



Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

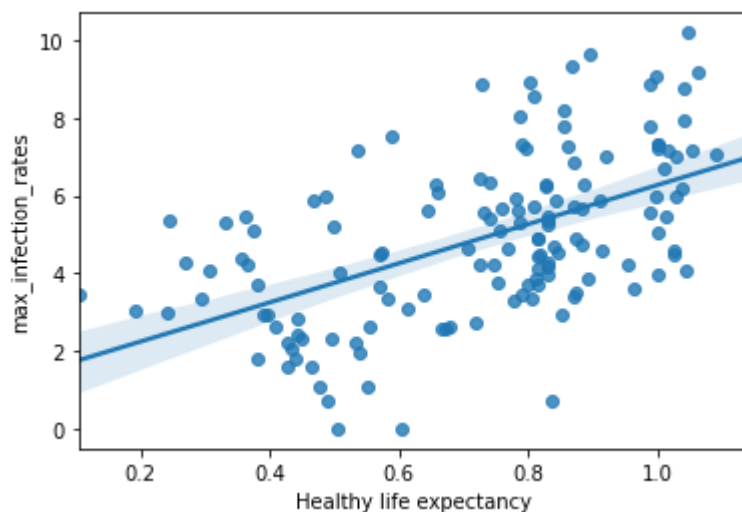
```
In [64]: x = data["Healthy life expectancy"]  
y = data["max_infection_rates"]  
sns.scatterplot(x,np.log(y))
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0xe391b98>



```
In [65]: sns.regplot(x, np.log(y))
```

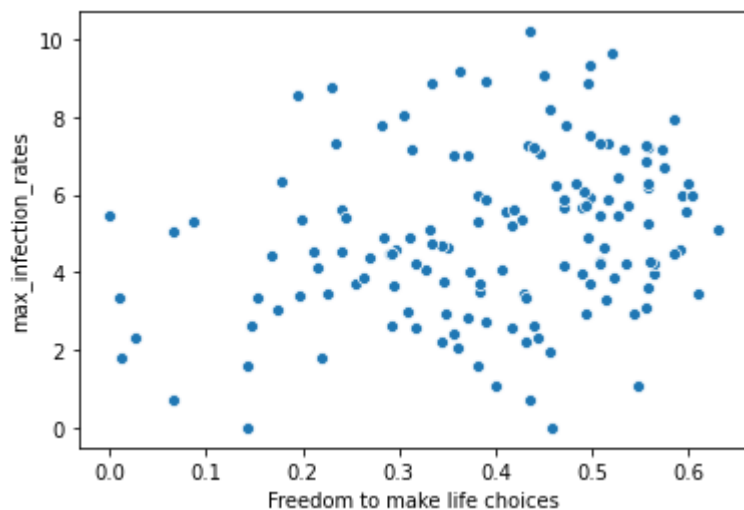
Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0xe43cfa0>



Task 5.4: Plotting Freedom to make life choices vs maximum Infection rate

```
In [66]: x = data["Freedom to make life choices"]  
y = data["max_infection_rates"]  
sns.scatterplot(x,np.log(y))
```

Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0xe406bf8>



```
In [67]: sns.regplot(x, np.log(y))
```

Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0xe4c5c88>

