Let's Import the modules

In [62]:
```python
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 [63]:
```python
corona_dataset_csv = pd.read_csv("Datasets/covid19_Confirmed_dataset.csv")
corona_dataset_csv.head()
```

Out[63]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/21/20	4/22/20	4/23/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	1092	1176	1279
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	609	634	663
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	2811	2910	3007
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	717	723	723
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	24	25	25

5 rows × 104 columns

Let's check the shape of the dataframe

In [64]:
```python
corona_dataset_csv.shape
```

Out[64]: (266, 104)

Task 2.2: Delete the useless columns

In [65]:
```python
corona_dataset_csv.drop(["Lat", "Long"], axis = 1, inplace = True)
```

In [66]:
```python
corona_dataset_csv.head(10)
```

Out[66]:

	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	...	4/21/20	4/22/20	4/23/20
0	NaN	Afghanistan	0	0	0	0	0	0	0	0	...	1092	1176	1279
1	NaN	Albania	0	0	0	0	0	0	0	0	...	609	634	663
2	NaN	Algeria	0	0	0	0	0	0	0	0	...	2811	2910	3007
3	NaN	Andorra	0	0	0	0	0	0	0	0	...	717	723	723
4	NaN	Angola	0	0	0	0	0	0	0	0	...	24	25	25
5	NaN	Antigua and Barbuda	0	0	0	0	0	0	0	0	...	23	24	24
6	NaN	Argentina	0	0	0	0	0	0	0	0	...	3031	3144	3435
7	NaN	Armenia	0	0	0	0	0	0	0	0	...	1401	1473	1523
8	Australian Capital Territory	Australia	0	0	0	0	0	0	0	0	...	104	104	104
9	New South Wales	Australia	0	0	0	0	3	4	4	4	...	2969	2971	2976

10 rows × 102 columns

Task 2.3: Aggregating the rows by the country

In [67]:
```python
corona_dataset_aggregated = corona_dataset_csv.groupby("Country/Region").sum()
```

In [68]:
```python
corona_dataset_aggregated.head()
```

Out[68]:

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	1/30/20	1/31/20	...	4/21/20	4/22/20	4/23/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1092	1176	1279
Albania	0	0	0	0	0	0	0	0	0	0	...	609	634	663
Algeria	0	0	0	0	0	0	0	0	0	0	...	2811	2910	3007
Andorra	0	0	0	0	0	0	0	0	0	0	...	717	723	723
Angola	0	0	0	0	0	0	0	0	0	0	...	24	25	25

5 rows × 100 columns

In [69]:
```python
corona_dataset_aggregated.shape
```

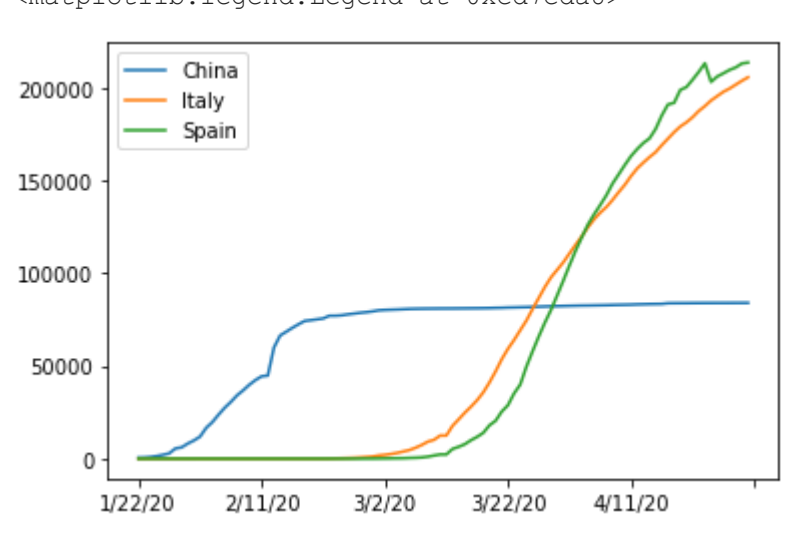
Out[69]: (187, 100)

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

visualization always helps for better understanding of our data.

In [70]:
```python
corona_dataset_aggregated.loc["China"].plot()
corona_dataset_aggregated.loc["Italy"].plot()
corona_dataset_aggregated.loc["Spain"].plot()
plt.legend()
```

Out[70]: <matplotlib.legend.Legend at 0xed7eda8>

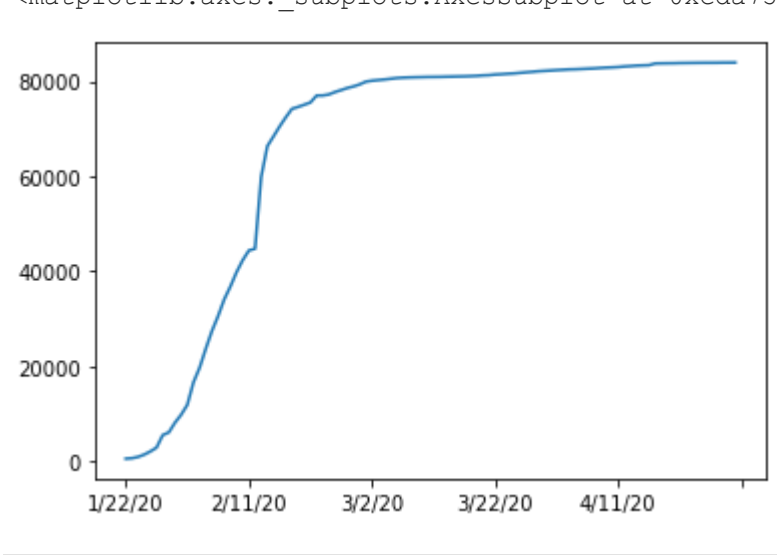


Task3: Calculating a good measure

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

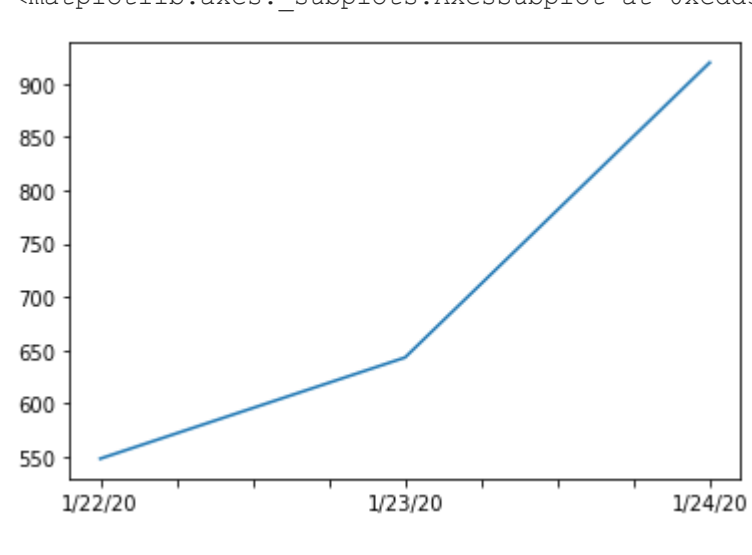
In [71]:
```python
corona_dataset_aggregated.loc['China'].plot()
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0xeda7550>



In [72]:
```python
corona_dataset_aggregated.loc["China"][:3].plot()
```

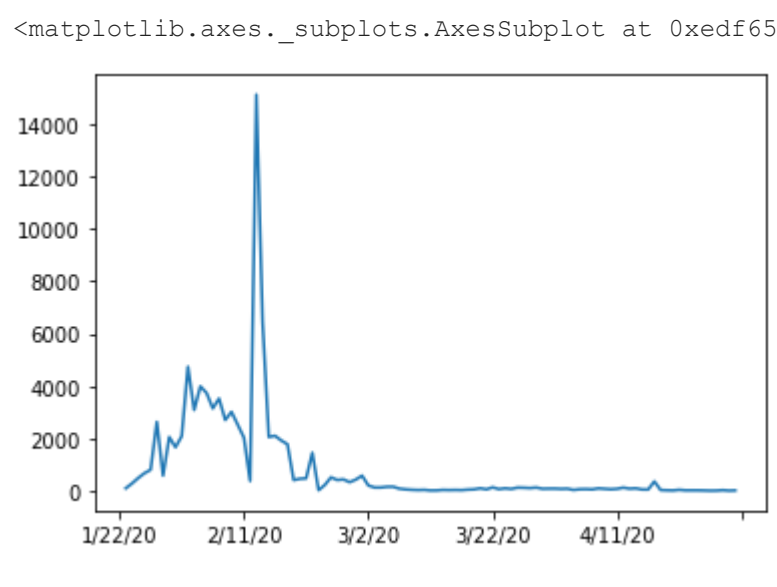
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0xedd38e0>



task 3.1: caculating the first derivative of the curve

In [73]:
```python
corona_dataset_aggregated.loc["China"].diff().plot()
```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0xedf65b0>



task 3.2: find maxmimum infection rate for China

In [74]:
```python
corona_dataset_aggregated.loc["China"].diff().max()
```

Out[74]: 15136.0

In [75]:
```python
corona_dataset_aggregated.loc["Italy"].diff().max()
```

Out[75]: 6557.0

In [76]:
```python
corona_dataset_aggregated.loc["Spain"].diff().max()
```

Out[76]: 9630.0

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

In [77]:
```python
countries = list(corona_dataset_aggregated.index)
max_infection_rate = []
for c in countries:
    max_infection_rate.append(corona_dataset_aggregated.loc[c].diff().max())

corona_dataset_aggregated["max_infection_rate"] = max_infection_rates
```

In [78]:
```python
corona_dataset_aggregated.head(10)
```

Out[78]:

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	1/30/20	1/31/20	...	4/22/20	4/23/20	4/24/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1176	1279	1351
Albania	0	0	0	0	0	0	0	0	0	0	...	634	663	678
Algeria	0	0	0	0	0	0	0	0	0	0	...	2910	3007	3127
Andorra	0	0	0	0	0	0	0	0	0	0	...	723	723	731
Angola	0	0	0	0	0	0	0	0	0	0	...	25	25	25
Antigua and Barbuda	0	0	0	0	0	0	0	0	0	0	...	24	24	24
Argentina	0	0	0	0	0	0	0	0	0	0	...	3144	3435	3607
Armenia	0	0	0	0	0	0	0	0	0	0	...	1473	1523	1596
Australia	0	0	0	0	4	5	5	6	9	9	...	6652	6662	6677
Austria	0	0	0	0	0	0	0	0	0	0	...	14925	15002	15071

10 rows × 101 columns

Task 3.4: create a new dataframe with only needed column

In [79]:
```python
corona_data = pd.DataFrame(corona_dataset_aggregated["max_infection_rate"])
```

In [80]:
```python
corona_data.head()
```

Out[80]:

Country/Region	max_infection_rate
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 [81]:
```python
happiness_report_csv = pd.read_csv("Datasets/worldwide_happiness_report.csv")
```

In [82]:
```python
happiness_report_csv.head()
```

Out[82]:

	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 [83]:
```python
useless_cols = ["Overall rank", "Score", "Generosity", "Perceptions of corruption"]
```

In [84]:
```python
happiness_report_csv .drop(useless_cols, axis = 1, inplace = True)
happiness_report_csv.head()
```

Out[84]:

	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 [85]:
```python
happiness_report_csv.set_index("Country or region", inplace = True)
```

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

Corona Dataset :

In [90]:
```python
corona_data.head()
```

Out[90]:

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

In [91]:
```python
corona_data.shape
```

Out[91]: (187, 1)

wolrd happiness report Dataset :

In [88]:
```python
happiness_report_csv.head()
```

Out[88]:

Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
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 [89]:
```python
happiness_report_csv.shape
```

Out[89]: (156, 4)

In [95]:
```python
data = corona_data.join(happiness_report_csv, how = "inner")
```

In [96]:
```python
data.head()
```

Out[96]:

	max_infection_rate	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 [97]:
```python
data.corr()
```

Out[97]:

	max_infection_rate	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
max_infection_rate	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 [98]:
```python
data.head()
```

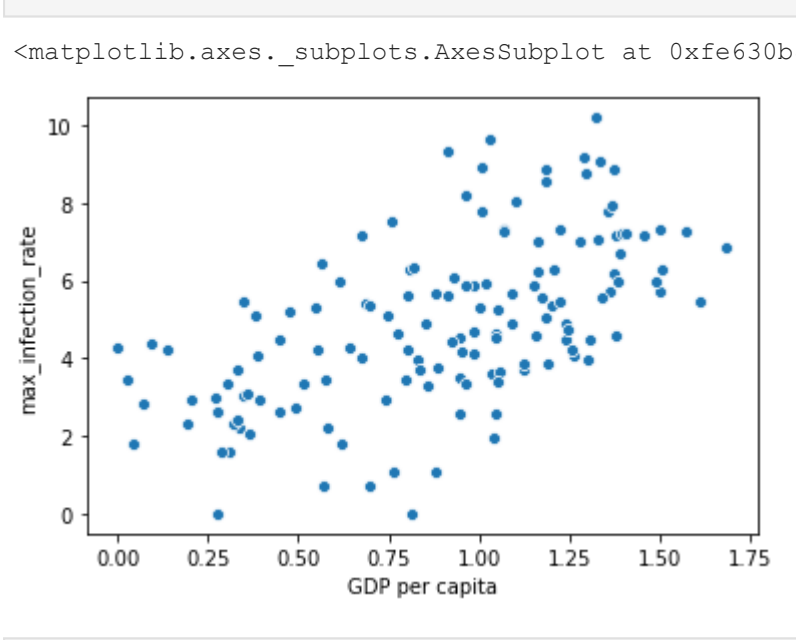
Out[98]:

	max_infection_rate	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

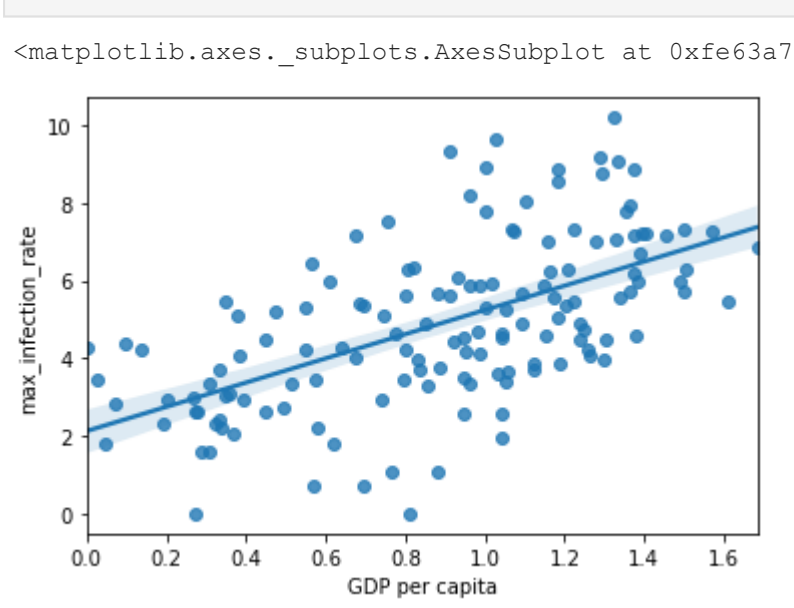
In [100…
```python
x = data["GDP per capita"]
y = data["max_infection_rate"]
sns.scatterplot(x, np.log(y))
```

Out[100… <matplotlib.axes._subplots.AxesSubplot at 0xfe630b8>



In [101…
```python
sns.regplot(x, np.log(y))
```

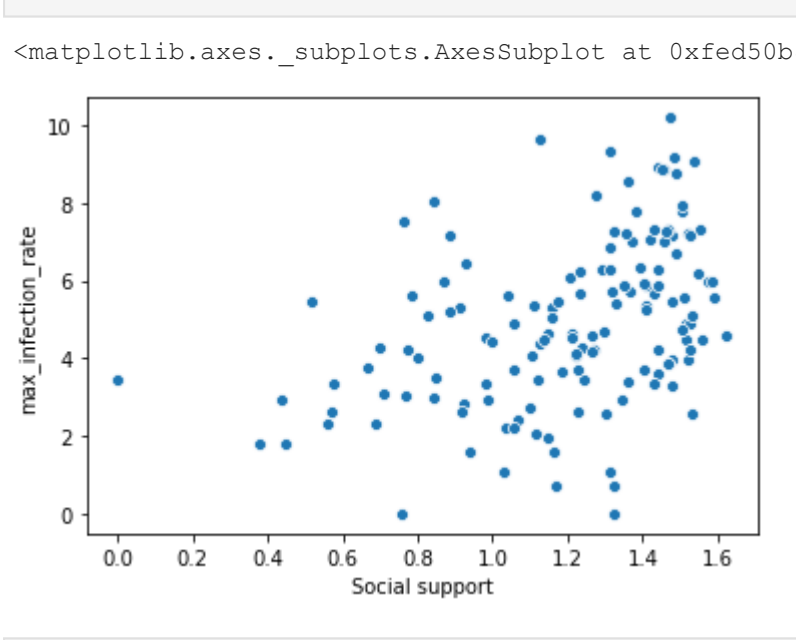
Out[101… <matplotlib.axes._subplots.AxesSubplot at 0xfe63a78>



Task 5.2: Plotting Social support vs maximum Infection rate

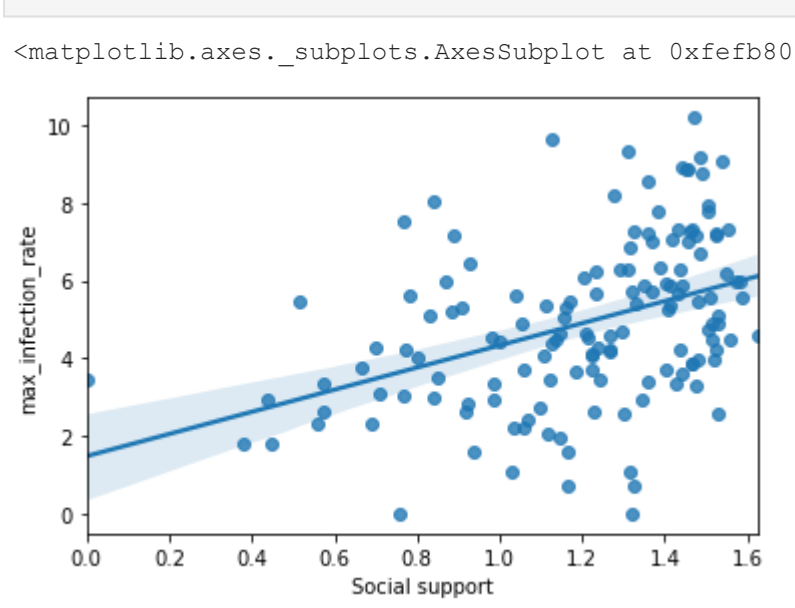
In [103…
```python
z = data["Social support"]
y = data["max_infection_rate"]
sns.scatterplot(z, np.log(y))
```

Out[103… <matplotlib.axes._subplots.AxesSubplot at 0xfed50b8>



In [104…
```python
sns.regplot(z, np.log(y))
```

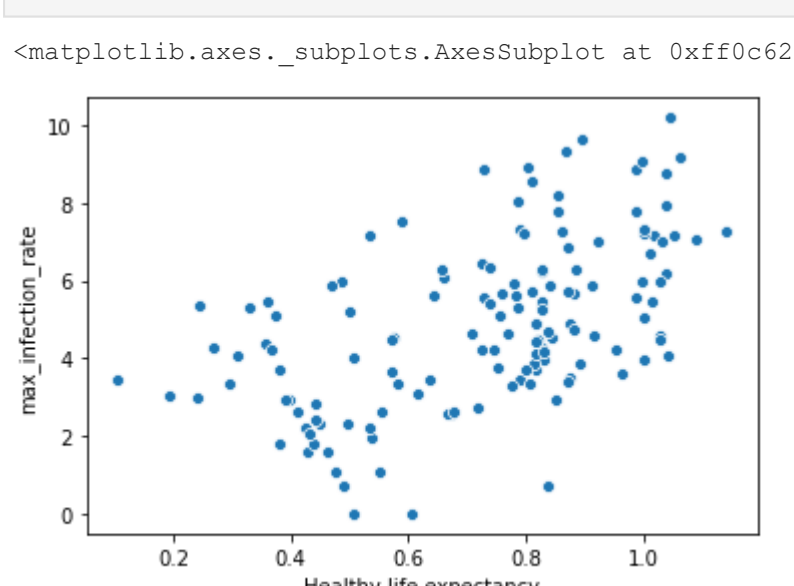
Out[104… <matplotlib.axes._subplots.AxesSubplot at 0xfefb808>



Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

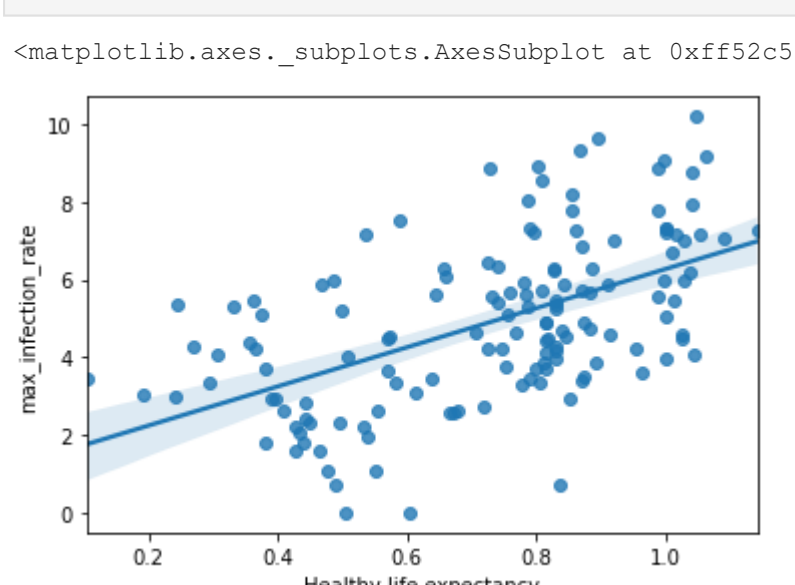
In [105…
```python
w = data["Healthy life expectancy"]
y = data["max_infection_rate"]
sns.scatterplot(w, np.log(y))
```

Out[105… <matplotlib.axes._subplots.AxesSubplot at 0xff0c628>



In [106…
```python
sns.regplot(w, np.log(y))
```

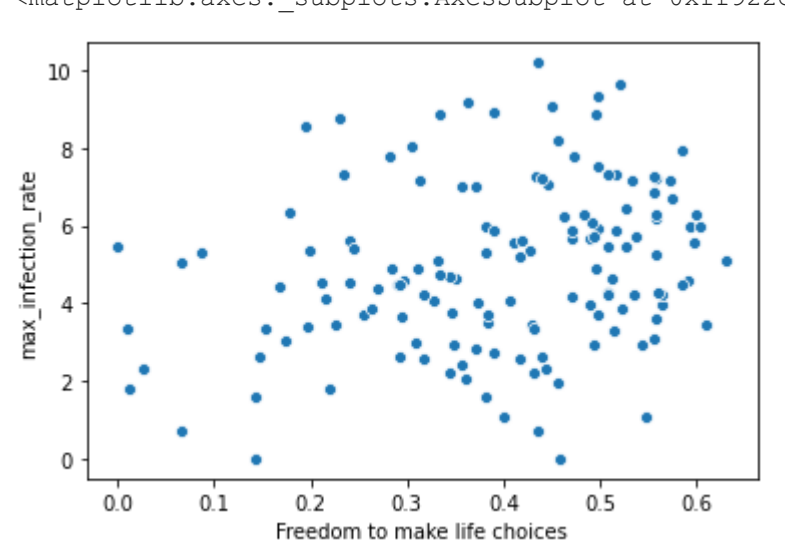
Out[106… <matplotlib.axes._subplots.AxesSubplot at 0xff52c58>



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

In [107…
```python
g = data["Freedom to make life choices"]
y = data["max_infection_rate"]
sns.scatterplot(g, np.log(y))
```

Out[107… <matplotlib.axes._subplots.AxesSubplot at 0xff922e0>



In [108…
```python
sns.regplot(g, np.log(y))
```

Out[108… <matplotlib.axes._subplots.AxesSubplot at 0xffb9070>