Importing the Modules

In [43]:
```
import pandas as pd
import numpy as nms
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing the dataset

In [7]:
```
covid19 = pd.read_csv("desktop/covid19_dataset.csv")
covid19.head(10)
```

Out[7]:

	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	4/24/20	4/25
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	1092	1176	1279	1351	14
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	609	634	663	678	7
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	2811	2910	3007	3127	32
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	717	723	723	731	7
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	24	25	25	25	
5	NaN	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0	...	23	24	24	24	
6	NaN	Argentina	-38.4161	-63.6167	0	0	0	0	0	0	...	3031	3144	3435	3607	37
7	NaN	Armenia	40.0691	45.0382	0	0	0	0	0	0	...	1401	1473	1523	1596	16
8	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0	...	104	104	104	105	1
9	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3	4	...	2969	2971	2976	2982	29

10 rows × 104 columns

Checking the shape

In [8]:
```
covid19.shape
```

Out[8]: (266, 104)

Deleing the useless columns

In [9]:
```
covid19.drop(["Lat","Long"],axis=1,inplace=True)
```

In [10]:
```
covid19.head(10)
```

Out[10]:

	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	4/24/20	4/25/20
0	NaN	Afghanistan	0	0	0	0	0	0	0	0	...	1092	1176	1279	1351	1463
1	NaN	Albania	0	0	0	0	0	0	0	0	...	609	634	663	678	712
2	NaN	Algeria	0	0	0	0	0	0	0	0	...	2811	2910	3007	3127	3256
3	NaN	Andorra	0	0	0	0	0	0	0	0	...	717	723	723	731	738
4	NaN	Angola	0	0	0	0	0	0	0	0	...	24	25	25	25	25
5	NaN	Antigua and Barbuda	0	0	0	0	0	0	0	0	...	23	24	24	24	24
6	NaN	Argentina	0	0	0	0	0	0	0	0	...	3031	3144	3435	3607	3780
7	NaN	Armenia	0	0	0	0	0	0	0	0	...	1401	1473	1523	1596	1677
8	Australian Capital Territory	Australia	0	0	0	0	0	0	0	0	...	104	104	104	105	106
9	New South Wales	Australia	0	0	0	0	3	4	4	4	...	2969	2971	2976	2982	2994

10 rows × 102 columns

In [ ]:
```
aggregating the rows by country
```

In [11]:
```
covid19_aggregated = covid19.groupby("Country/Region").sum()
covid19_aggregated.head()
```

Out[11]:

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	4/24/20	4/25/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1092	1176	1279	1351	1463
Albania	0	0	0	0	0	0	0	0	0	0	...	609	634	663	678	712
Algeria	0	0	0	0	0	0	0	0	0	0	...	2811	2910	3007	3127	3256
Andorra	0	0	0	0	0	0	0	0	0	0	...	717	723	723	731	738
Angola	0	0	0	0	0	0	0	0	0	0	...	24	25	25	25	25

5 rows × 100 columns

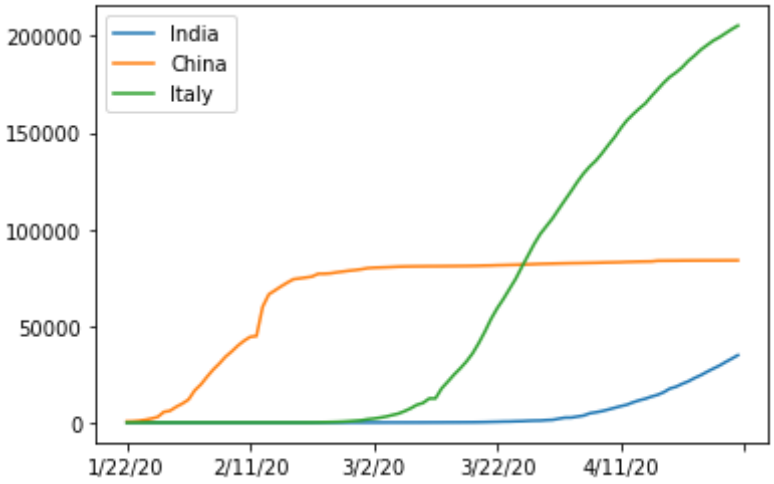
In [12]:
```
covid19_aggregated.shape
```

Out[12]:
```
(187, 100)
```

Visualizing data related to countries

In [13]:
```
covid19_aggregated.loc["India"].plot()
covid19_aggregated.loc["China"].plot()
covid19_aggregated.loc["Italy"].plot()
plt.legend()
```

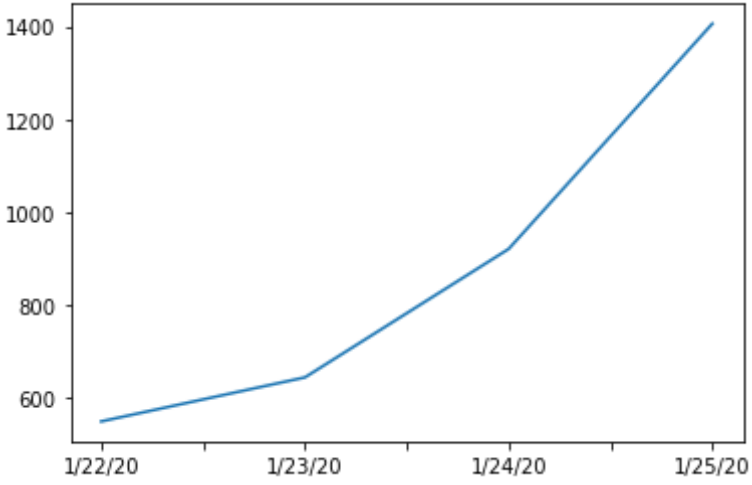
Out[13]:
```
<matplotlib.legend.Legend at 0x1f10bd94e80>
```



To calculate a good measure

In [14]:
```
covid19_aggregated.loc["China"][:4].plot()
```

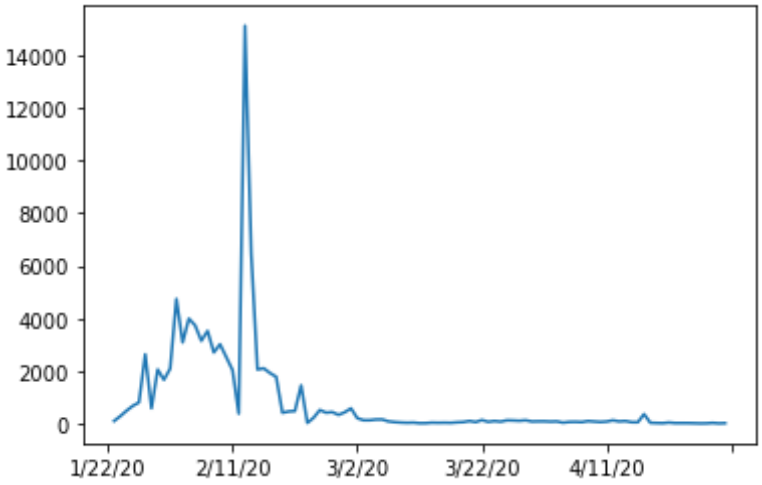
Out[14]: <AxesSubplot:>



Calculating and plotting the first derivative

In [15]:
```
covid19_aggregated.loc["China"].diff().plot()
```

Out[15]: <AxesSubplot:>



Finding maximum infection rate

In [16]:
```
covid19_aggregated.loc["China"].diff().max()
```

Out[16]: 15136.0

In [17]:
```
covid19_aggregated.loc["India"].diff().max()
```

Out[17]: 1893.0

In [18]:
```
covid19_aggregated.loc["Italy"].diff().max()
```

Out[18]: 6557.0

Finding maximum infection rate for all the countries

In [19]:
```
countries = list(covid19_aggregated.index)
max_infection_rates = []
for c in countries :
    max_infection_rates.append(covid19_aggregated.loc[c].diff().max())
covid19_aggregated["max_infection_rate"] = max_infection_rates
```

In [20]:
```
covid19_aggregated.head()
```

Out[20]:

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	4/25/20	4/26/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1176	1279	1351	1463	1531
Albania	0	0	0	0	0	0	0	0	0	0	...	634	663	678	712	726
Algeria	0	0	0	0	0	0	0	0	0	0	...	2910	3007	3127	3256	3382
Andorra	0	0	0	0	0	0	0	0	0	0	...	723	723	731	738	738
Angola	0	0	0	0	0	0	0	0	0	0	...	25	25	25	25	26

5 rows × 101 columns

Creating a new dataframe with only needed columns

In [21]:
```
covid19 = pd.DataFrame(covid19_aggregated["max_infection_rate"])
```

In [22]:
```
covid19.head()
```

Out[22]:

Country/Region	max_infection_rate
Afghanistan	232.0
Albania	34.0

Country/Region	max_infection_rate
Algeria	199.0
Andorra	43.0
Angola	5.0

Importing a new dataset

In [23]:
```
happyreport = pd.read_csv("Desktop/worldwide_happiness_report.csv")
```

In [24]:
```
happyreport.head()
```

Out[24]:

	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

Removing useless columns

In [25]:
```
useless_cols = ["Overall rank","Score","Generosity","Perceptions of corruption"]
```

In [26]:
```
happyreport.drop(useless_cols,axis=1,inplace=True)
happyreport.head()
```

Out[26]:

	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

	Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
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

Changing the indices of the dataframe

In [27]:
```
happyreport.set_index("Country or region",inplace=True)
```

In [28]:
```
happyreport.head()
```

Out[28]:

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

Joining the two datasets

COVID19 DATASET

In [29]:
```
covid19.head()
```

Out[29]:

	max_infection_rate

Country/Region	max_infection_rate

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

In [30]:
```
covid19.shape
```

Out[30]: (187, 1)

WORLD HAPPINESS DATASET

In [31]:
```
happyreport.head()
```

Out[31]:

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 [32]:
```
happyreport.shape
```

Out[32]: (156, 4)

Using Join method

In [33]:
```
newdata = covid19.join(happyreport,how="inner")
newdata.head()
```

Out[33]:

	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

Correlation matrix

In [34]:
```
newdata.corr()
```

Out[34]:

	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

Visualisation of the results

In [42]:
```
newdata.head()
```

Out[42]:

	max_infection_rate	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices

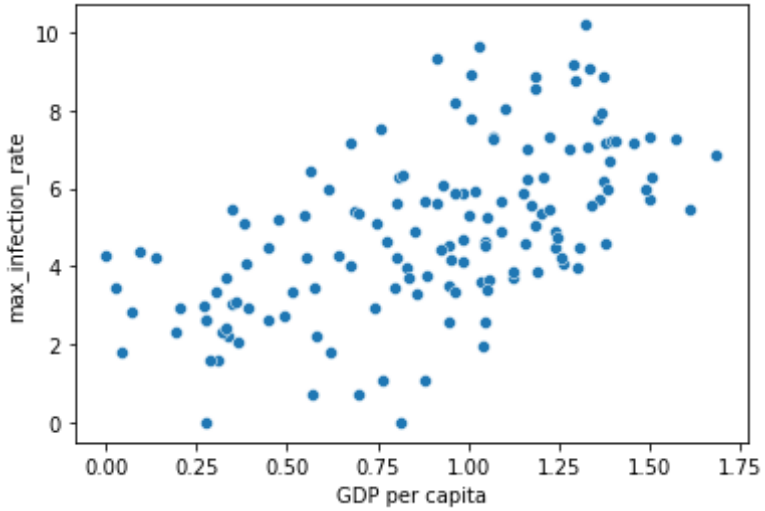
	max_infection_rate	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

Plotting GDP vs Maximum infection rate

In [35]:
```
x = newdata["GDP per capita"]
y = newdata["max_infection_rate"]
sns.scatterplot(x, nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[35]: <AxesSubplot:xlabel='GDP per capita', ylabel='max_infection_rate'>

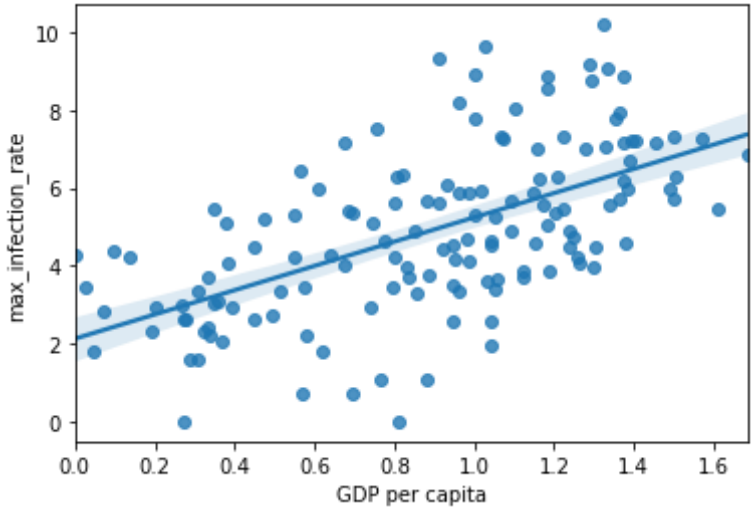


Using regression plot

In [36]:
```
sns.regplot(x,nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[36]: <AxesSubplot:xlabel='GDP per capita', ylabel='max_infection_rate'>

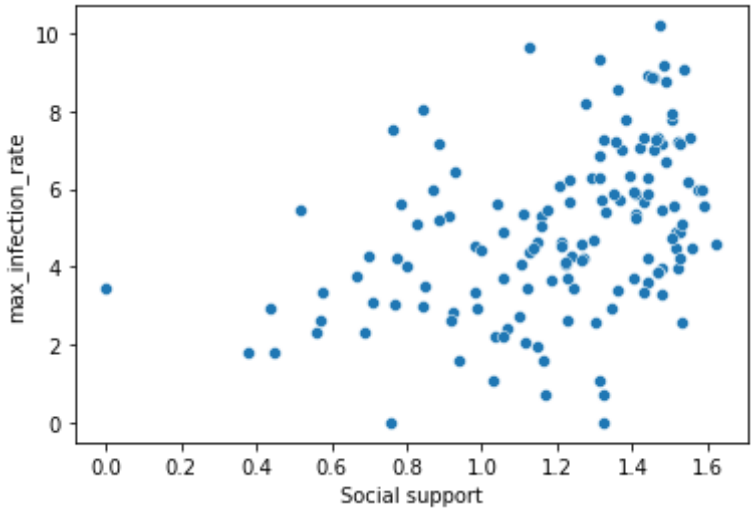


Plotting social support vs maximum infection rate

In [37]:
```
x = newdata["Social support"]
y = newdata["max_infection_rate"]
sns.scatterplot(x,nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[37]: <AxesSubplot:xlabel='Social support', ylabel='max_infection_rate'>

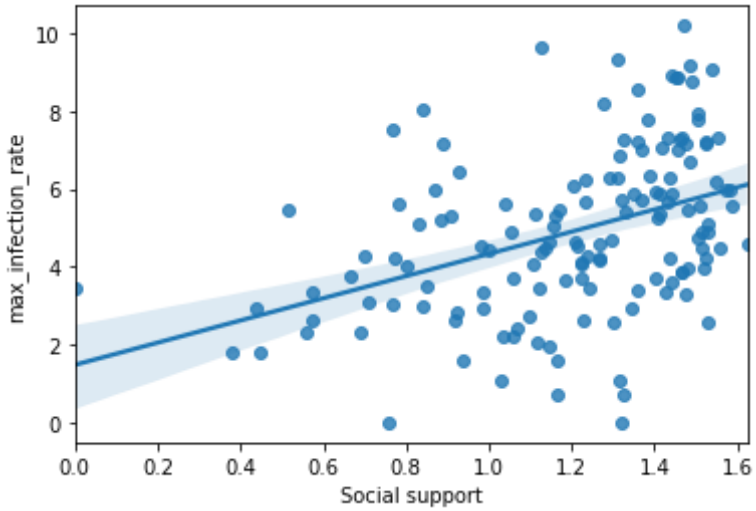


Regression plot

In [38]:
```
sns.regplot(x,nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[38]:
```
<AxesSubplot:xlabel='Social support', ylabel='max_infection_rate'>
```

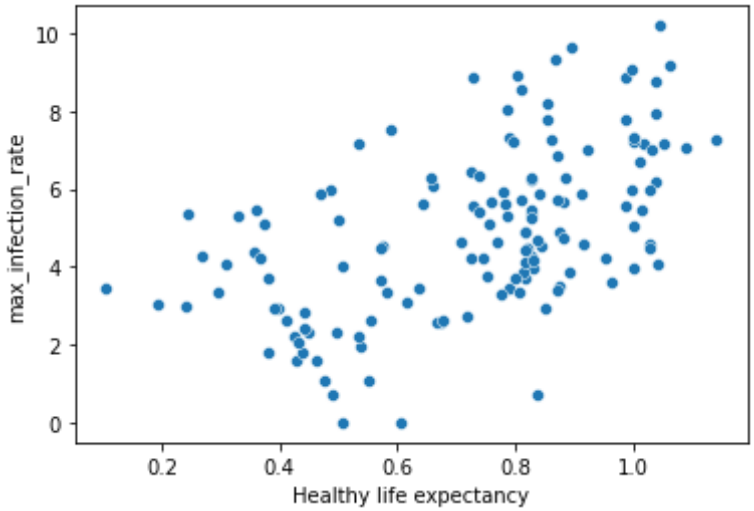


Plotting healthy life expectancy vs maximum infection rate

In [39]:
```
x = newdata["Healthy life expectancy"]
y = newdata["max_infection_rate"]
sns.scatterplot(x,nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[39]: <AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rate'>

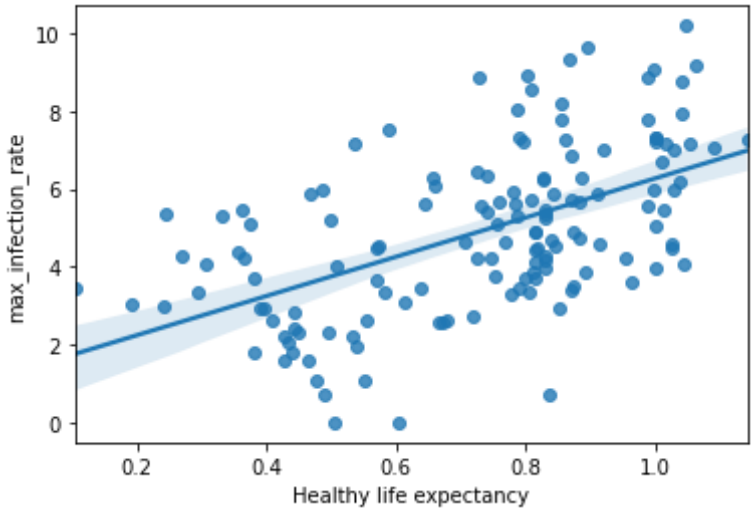


Regression plot

In [40]:
```
sns.regplot(x,nms.log(y))
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as k
eyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[40]:
```
<AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rate'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: