College code: 4212
Register num: 421221243041

COVID-19 CASES ANALYSIS

DATA ANALYTICS WITH COGNOS:GROUP2

PHASE:3

This phase involves in designing of the steps that defining in each phase of the previous documentation this involves importing necessary functions, data processing and so on in this phase we have to begin our project by loading and preprocessing the dataset.

The IBM suggests using the jupyter notebook for loading and preprocess the dataset:

Here for this project title we need to define the loading the libraries, understand the data and visualize the missing values.

For this certain inputs are defined for this project.in this phase each of the input lines of the project is given as follows:

IBM NAAN MUDHULVAN PHASE3

phase3

October 17, 2023

# Welcome to Covid19 Data Analysis Notebook

Let's Import the modules

```
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.

```
corona_dataset_csv = pd.read_csv('Covid19_Confirmed_dataset.csv')
corona_dataset_csv.head(10)
#We will notice data is from 22 January 2020 to 30 April 2020
```

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

   1/22/20  1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  ...  4/21/20  \
0        0        0        0        0        0        0  ...     1092
1        0        0        0        0        0        0  ...      609
2        0        0        0        0        0        0  ...     2811
3        0        0        0        0        0        0  ...      717
4        0        0        0        0        0        0  ...       24
5        0        0        0        0        0        0  ...       23
6        0        0        0        0        0        0  ...     3031
7        0        0        0        0        0        0  ...     1401
8        0        0        0        0        0        0  ...      104
9        0        0        0        0        3        4  ...     2969
```

```
   4/22/20  4/23/20  4/24/20  4/25/20  4/26/20  4/27/20  4/28/20  4/29/20  \
0     1176     1279     1351     1463     1531     1703     1828     1939   
1      634      663      678      712      726      736      750      766   
2     2910     3007     3127     3256     3382     3517     3649     3848   
3      723      723      731      738      738      743      743      743   
4       25       25       25       25       26       27       27       27   
5       24       24       24       24       24       24       24       24   
6     3144     3435     3607     3780     3892     4003     4127     4285   
7     1473     1523     1596     1677     1746     1808     1867     1932   
8      104      104      105      106      106      106      106      106   
9     2971     2976     2982     2994     3002     3004     3016     3016   

   4/30/20  
0     2171  
1      773  
2     4006  
3      745  
4       27  
5       24  
6     4428  
7     2066  
8      106  
9     3025  

[10 rows x 104 columns]
```

Let's check the shape of the dataframe

```
corona_dataset_csv.shape    #Tuple with 266 rows and 104 columns
```

```
(266, 104)
```

```
columns = corona_dataset_csv.columns
columns
```

```
Index(['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', '4/25/20', '4/26/20',
       '4/27/20', '4/28/20', '4/29/20', '4/30/20'],
      dtype='object', length=104)
```

Task 2.2: Delete the useless columns

```
#Latitude and Longitude are not important features for us here
corona_dataset_csv.drop(["Lat",
                        "Long"],
                        axis=1,
                        #default value, annotation axis=0 which is equal to rows
                        inplace = True   #will change the corona dataset too
                        )
```

```
corona_dataset_csv.head(10)

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

   1/24/20  1/25/20  1/26/20  1/27/20  1/28/20  1/29/20  ...  4/21/20  \
0        0        0        0        0        0        0  ...     1092
1        0        0        0        0        0        0  ...      609
2        0        0        0        0        0        0  ...     2811
3        0        0        0        0        0        0  ...      717
4        0        0        0        0        0        0  ...       24
5        0        0        0        0        0        0  ...       23
6        0        0        0        0        0        0  ...     3031
7        0        0        0        0        0        0  ...     1401
8        0        0        0        0        0        0  ...      104
9        0        0        3        4        4        4  ...     2969

   4/22/20  4/23/20  4/24/20  4/25/20  4/26/20  4/27/20  4/28/20  4/29/20  \
0     1176     1279     1351     1463     1531     1703     1828     1939
1      634      663      678      712      726      736      750      766
2     2910     3007     3127     3256     3382     3517     3649     3848
3      723      723      731      738      738      743      743      743
4       25       25       25       25       26       27       27       27
5       24       24       24       24       24       24       24       24
6     3144     3435     3607     3780     3892     4003     4127     4285
7     1473     1523     1596     1677     1746     1808     1867     1932
8      104      104      105      106      106      106      106      106
9     2971     2976     2982     2994     3002     3004     3016     3016

   4/30/20
0     2171
1      773
2     4006
3      745
4       27
5       24
6     4428
7     2066
8      106
9     3025
```

```
[10 rows x 102 columns]
```

Task 2.3: Aggregating the rows by the country

```
corona_dataset_aggregated = corona_dataset_csv.groupby("Country/Region").sum()

corona_dataset_aggregated.head()
#After aggregation, the index of the df is the column at which we aggregated
```

```
                1/22/20  1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  1/28/20  \
Country/Region
Afghanistan           0        0        0        0        0        0        0
Albania               0        0        0        0        0        0        0
Algeria               0        0        0        0        0        0        0
Andorra               0        0        0        0        0        0        0
Angola                0        0        0        0        0        0        0

                1/29/20  1/30/20  1/31/20  ...  4/21/20  4/22/20  4/23/20  \
Country/Region                             ...
Afghanistan           0        0        0  ...     1092     1176     1279
Albania               0        0        0  ...      609      634      663
Algeria               0        0        0  ...     2811     2910     3007
Andorra               0        0        0  ...      717      723      723
Angola                0        0        0  ...       24       25       25

                4/24/20  4/25/20  4/26/20  4/27/20  4/28/20  4/29/20  4/30/20
Country/Region
Afghanistan        1351     1463     1531     1703     1828     1939     2171
Albania             678      712      726      736      750      766      773
Algeria            3127     3256     3382     3517     3649     3848     4006
Andorra             731      738      738      743      743      743      745
Angola               25       25       26       27       27       27       27

[5 rows x 100 columns]
```

```
corona_dataset_aggregated.shape
#we have 187 countries, 100 dates
```

```
(187, 100)
```

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

visualization always helps for better understanding of our data.

```
corona_dataset_aggregated.loc["China"]
#will return pandas series
```

```
1/22/20      548
1/23/20      643
1/24/20      920
1/25/20     1406
```

```
1/26/20      2075
1/27/20      2877
1/28/20      5509
1/29/20      6087
1/30/20      8141
1/31/20      9802
2/1/20      11891
2/2/20      16630
2/3/20      19716
2/4/20      23707
2/5/20      27440
2/6/20      30587
2/7/20      34110
2/8/20      36814
2/9/20      39829
2/10/20     42354
2/11/20     44386
2/12/20     44759
2/13/20     59895
2/14/20     66358
2/15/20     68413
2/16/20     70513
2/17/20     72434
2/18/20     74211
2/19/20     74619
2/20/20     75077
            ...
4/1/20      82361
4/2/20      82432
4/3/20      82511
4/4/20      82543
4/5/20      82602
4/6/20      82665
4/7/20      82718
4/8/20      82809
4/9/20      82883
4/10/20     82941
4/11/20     83014
4/12/20     83134
4/13/20     83213
4/14/20     83306
4/15/20     83356
4/16/20     83403
4/17/20     83760
4/18/20     83787
4/19/20     83805
4/20/20     83817
4/21/20     83853
4/22/20     83868
4/23/20     83884
```

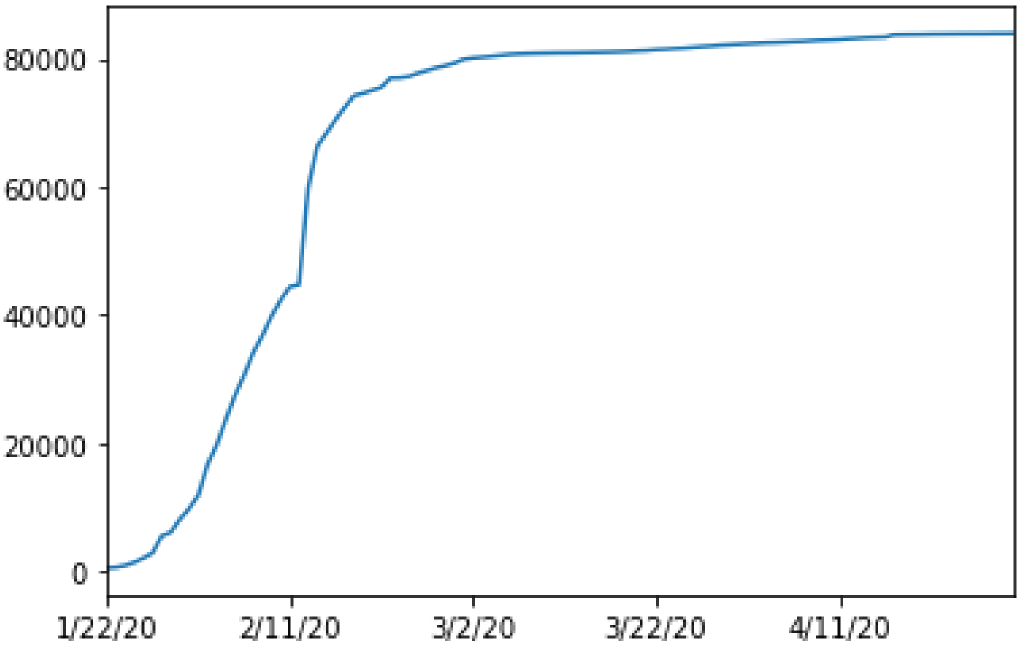
```
4/24/20    83899
4/25/20    83909
4/26/20    83912
4/27/20    83918
4/28/20    83940
4/29/20    83944
4/30/20    83956
Name: China, Length: 100, dtype: int64
```

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.

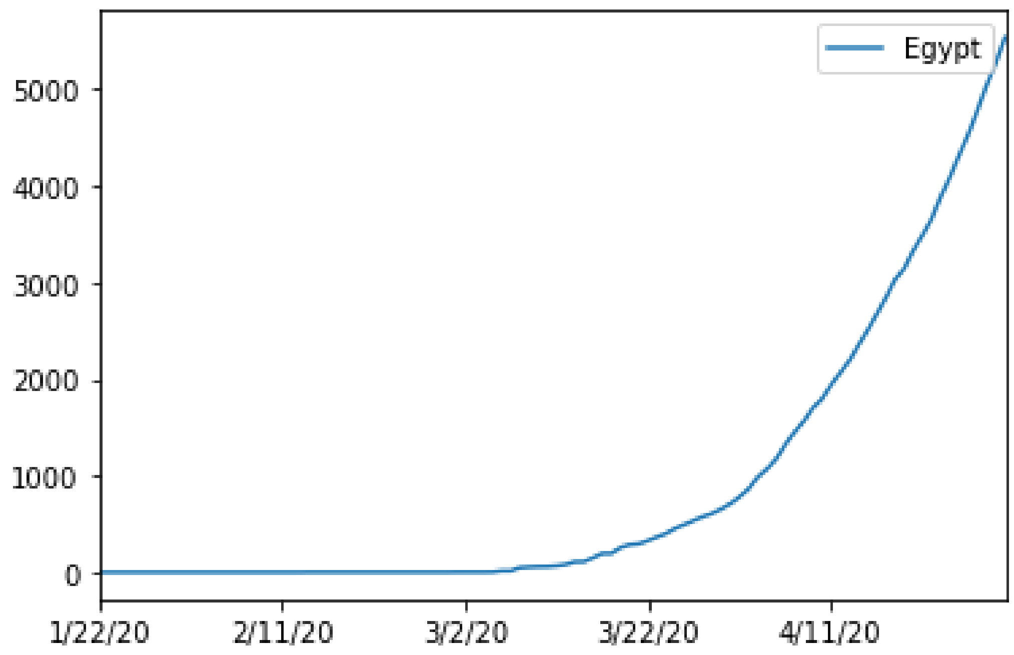
```
corona_dataset_aggregated.loc['China'].plot()
#will plot the values on different date
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4c9fc828>
```



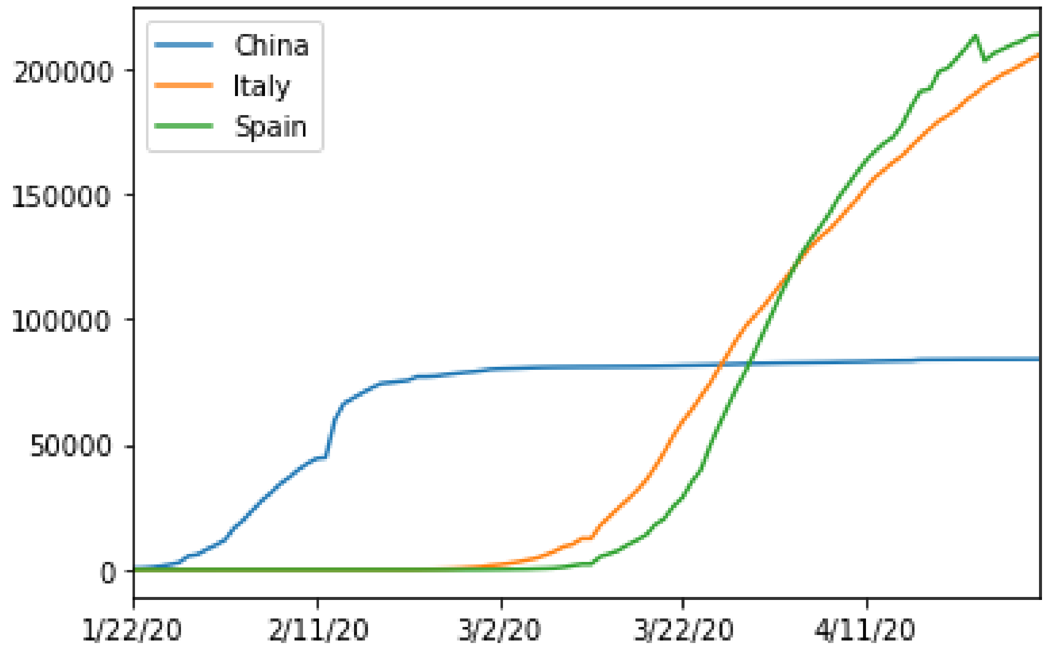
```
corona_dataset_aggregated.loc['Egypt'].plot()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x20d4cd43ba8>
```



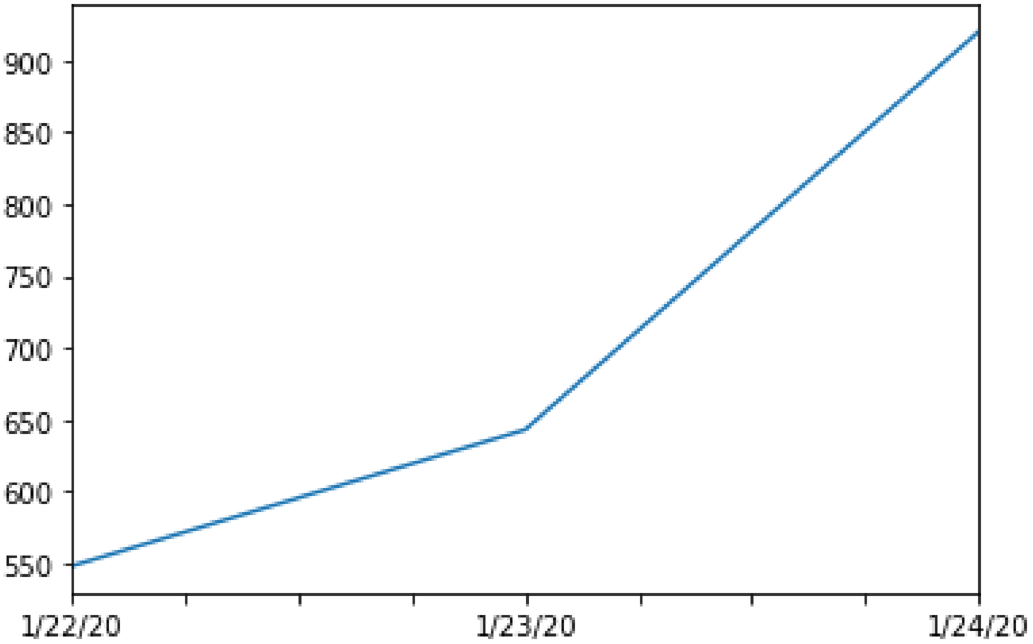
```
corona_dataset_aggregated.loc['China'].plot()
corona_dataset_aggregated.loc['Italy'].plot()
corona_dataset_aggregated.loc['Spain'].plot()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x20d4cd9fa58>
```



```
#Spread of the virus in China for the first 3 dates only
corona_dataset_aggregated.loc['China'][:3].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4ce2fdd8>
```



In the 1st 24 hrs, an increase in case (550 to 650) by 100

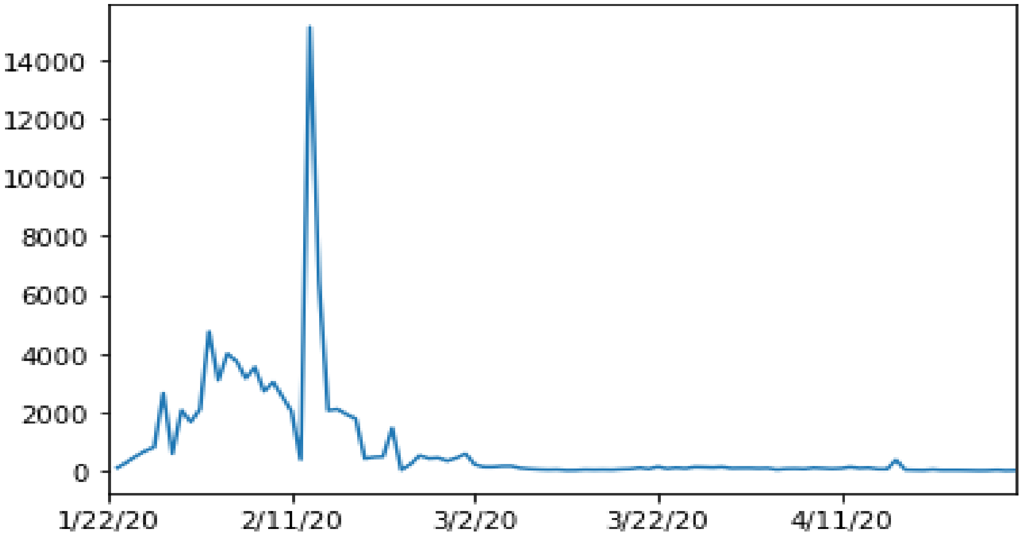
In the 2nd 24 hrs, an increase (650 to 900) by 250

We want to find a measure for new cases, so either say average or maximum number of new cases.

task 3.1: caculating the first derivative of the curve

```
corona_dataset_aggregated.loc["China"].diff().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4ce9fda0>
```



This plot shows us the change in infection rate day by day and what we are looking for is the maximum number.

task 3.2: find maxmimum infection rate for China

```
corona_dataset_aggregated.loc["China"].diff().max()
#In only 24 hrs, the difference was 15136
```

```
15136.0
```

```
corona_dataset_aggregated.loc["Italy"].diff().max()
#In only 24 hrs, the difference was 6557
```

```
6557.0
```

```
corona_dataset_aggregated.loc["Spain"].diff().max()
#In only 24 hrs, the difference was 9630
```

```
9630.0
```

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

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

```
[232.0,
 34.0,
 199.0,
 43.0,
 5.0,
 6.0,
 291.0,
 134.0,
 497.0,
 1321.0,
 105.0,
 7.0,
 301.0,
 641.0,
 12.0,
 1485.0,
 2454.0,
 4.0,
 19.0,
 1.0,
 104.0,
 92.0,
 7.0,
 7502.0,
 26.0,
 137.0,
 41.0,
 21.0,
 6.0,
 45.0,
 31.0,
 203.0,
 2778.0,
 31.0,
 21.0,
 1138.0,
 15136.0,
 353.0,
 1.0,
 57.0,
 81.0,
 37.0,
 113.0,
 96.0,
 63.0,
 58.0,
 381.0,
 391.0,
 99.0,
 156.0,
 5.0,
 371.0,
 11536.0,
 269.0,
 32.0,
 130.0,
 7.0,
 134.0,
 20.0,
 9.0,
 5.0,
 267.0,
 26849.0,
 38.0,
 5.0,
 42.0,
 6933.0,
 403.0,
 156.0,
 6.0,
 68.0,
 167.0,
 132.0,
 12.0,
 10.0,
 3.0,
 72.0,
 210.0,
 99.0,
 1893.0,
 436.0,
 3186.0,
 91.0,
 1515.0,
 1131.0,
 6557.0,
 52.0,
 1161.0,
 40.0,
 264.0,
 29.0,
 851.0,
 289.0,
 300.0,
 69.0,
 3.0,
 48.0,
 61.0,
 17.0,
 13.0,
 21.0,
 90.0,
 234.0,
 7.0,
 14.0,
 10.0,
 235.0,
 190.0,
 58.0,
 52.0,
 2.0,
 41.0,
 1425.0,
 222.0,
 12.0,
 13.0,
 30.0,
 281.0,
 19.0,
 3.0,
 14.0,
 1346.0,
 89.0,
 2.0,
 69.0,
 208.0,
 107.0,
 386.0,
 144.0,
 1292.0,
 357.0,
 5.0,
 27.0,
 3683.0,
 538.0,
 545.0,
 1516.0,
 957.0,
 523.0,
 7099.0,
 22.0,
 5.0,
 6.0,
 4.0,
 54.0,
 6.0,
 1351.0,
 87.0,
 2379.0,
 2.0,
 20.0,
 1426.0,
 114.0,
 70.0,
 73.0,
 354.0,
 28.0,
 9630.0,
 65.0,
 67.0,
 3.0,
 812.0,
 1321.0,
 6.0,
 27.0,
 15.0,
 181.0,
 188.0,
 10.0,
 14.0,
 40.0,
 82.0,
 5138.0,
 36188.0,
 11.0,
 578.0,
 552.0,
 8733.0,
 48.0,
 167.0,
 29.0,
 19.0,
 66.0,
 4.0,
 5.0,
 9.0,
 8.0]
```

```
corona_dataset_aggregated["max_infection_rates"] = max_infection_rates
```

```
corona_dataset_aggregated.head()
```

```
                1/22/20  1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  1/28/20  \
Country/Region
Afghanistan           0        0        0        0        0        0        0
Albania               0        0        0        0        0        0        0
Algeria               0        0        0        0        0        0        0
Andorra               0        0        0        0        0        0        0
Angola                0        0        0        0        0        0        0

                1/29/20  1/30/20  1/31/20  ...  4/22/20  4/23/20  4/24/20  \
Country/Region                             ...
Afghanistan           0        0        0  ...     1176     1279     1351
Albania               0        0        0  ...      634      663      678
Algeria               0        0        0  ...     2910     3007     3127
Andorra               0        0        0  ...      723      723      731
Angola                0        0        0  ...       25       25       25

                4/25/20  4/26/20  4/27/20  4/28/20  4/29/20  4/30/20  \
Country/Region
Afghanistan        1463     1531     1703     1828     1939     2171
Albania             712      726      736      750      766      773
Algeria            3256     3382     3517     3649     3848     4006
Andorra             738      738      743      743      743      745
Angola               25       26       27       27       27       27

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

[5 rows x 101 columns]
```

Task 3.4: create a new dataframe with only needed column

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

corona_data.head()

                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

```
happiness_report_csv = pd.read_csv("worldwide_happiness_report.csv")

happiness_report_csv.head()

   Overall rank Country or region  Score  GDP per capita  Social support  \
0             1           Finland  7.769           1.340           1.587
1             2           Denmark  7.600           1.383           1.573
2             3            Norway  7.554           1.488           1.582
3             4           Iceland  7.494           1.380           1.624
4             5       Netherlands  7.488           1.396           1.522

   Healthy life expectancy  Freedom to make life choices  Generosity  \
0                    0.986                         0.596       0.153
1                    0.996                         0.592       0.252
2                    1.028                         0.603       0.271
3                    1.026                         0.591       0.354
4                    0.999                         0.557       0.322

   Perceptions of corruption
0                      0.393
1                      0.410
2                      0.341
3                      0.118
4                      0.298
```

Task 4.2: let's drop the useless columns

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

```
happiness_report_csv.drop(useless_cols, axis=1, inplace=True)
happiness_report_csv.head()

  Country or region  GDP per capita  Social support  Healthy life expectancy  \
0           Finland           1.340           1.587                    0.986
1           Denmark           1.383           1.573                    0.996
2            Norway           1.488           1.582                    1.028
3           Iceland           1.380           1.624                    1.026
4       Netherlands           1.396           1.522                    0.999

   Freedom to make life choices
0                         0.596
1                         0.592
2                         0.603
3                         0.591
4                         0.557
```

Task 4.3: changing the indices of the dataframe

```
happiness_report_csv.set_index("Country or region", inplace=True)

happiness_report_csv.head()

                   GDP per capita  Social support  Healthy life expectancy  \
Country or region
Finland                     1.340           1.587                    0.986
Denmark                     1.383           1.573                    0.996
Norway                      1.488           1.582                    1.028
Iceland                     1.380           1.624                    1.026
Netherlands                 1.396           1.522                    0.999

                   Freedom to make life choices
Country or region
Finland                                   0.596
Denmark                                   0.592
Norway                                    0.603
Iceland                                   0.591
Netherlands                               0.557
```

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

Corona Dataset :

```
corona_data.head()

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

```
corona_data.shape    #Tuple with 187 countries
```

```
(187, 1)
```

wolrd happiness report Dataset :

```
happiness_report_csv.head()
```

```
                   GDP per capita  Social support  Healthy life expectancy  \
Country or region
Finland                     1.340           1.587                    0.986
Denmark                     1.383           1.573                    0.996
Norway                      1.488           1.582                    1.028
Iceland                     1.380           1.624                    1.026
Netherlands                 1.396           1.522                    0.999

                   Freedom to make life choices
Country or region
Finland                                   0.596
Denmark                                   0.592
Norway                                    0.603
Iceland                                   0.591
Netherlands                               0.557
```

```
happiness_report_csv.shape    #156 countries, less than corona data
```

```
(156, 4)
```

```
#Inner join
data = corona_data.join(happiness_report_csv,
                        how = "inner"    #method/type of join
                        )
data.head()
```

```
             max_infection_rates  GDP per capita  Social support  \
Afghanistan                232.0           0.350           0.517
Albania                     34.0           0.947           0.848
Algeria                    199.0           1.002           1.160
Argentina                  291.0           1.092           1.432
Armenia                    134.0           0.850           1.055

             Healthy life expectancy  Freedom to make life choices
Afghanistan                    0.361                         0.000
Albania                        0.874                         0.383
Algeria                        0.785                         0.086
Argentina                      0.881                         0.471
Armenia                        0.815                         0.283
```

Task 4.5: correlation matrix

```
data.corr()
```

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

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

                              Freedom to make life choices  
max_infection_rates                               0.078196  
GDP per capita                                    0.394603  
Social support                                    0.456246  
Healthy life expectancy                           0.427892  
Freedom to make life choices                      1.000000  
```

There is +ve correlation between max_infection-rate and all other features

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

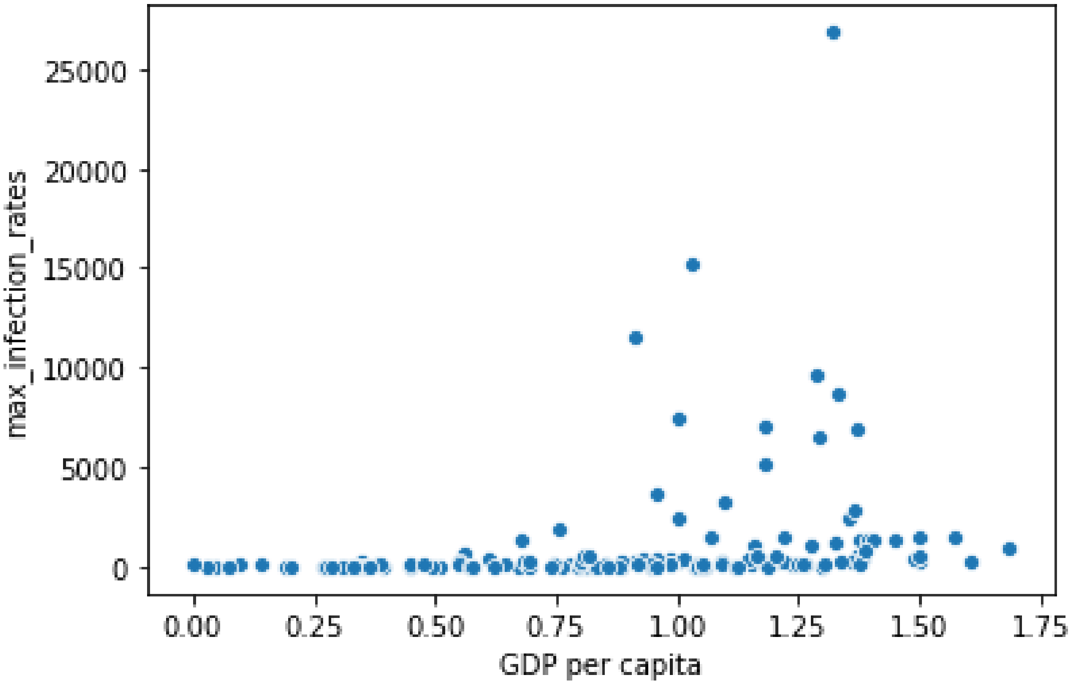
```
data.head()
```

```
             max_infection_rates  GDP per capita  Social support  \
Afghanistan                232.0           0.350           0.517   
Albania                     34.0           0.947           0.848   
Algeria                    199.0           1.002           1.160   
Argentina                  291.0           1.092           1.432   
Armenia                    134.0           0.850           1.055   

             Healthy life expectancy  Freedom to make life choices  
Afghanistan                    0.361                         0.000  
Albania                        0.874                         0.383  
Algeria                        0.785                         0.086  
Argentina                      0.881                         0.471  
Armenia                        0.815                         0.283  
```

Task 5.1: Plotting GDP vs maximum Infection rate

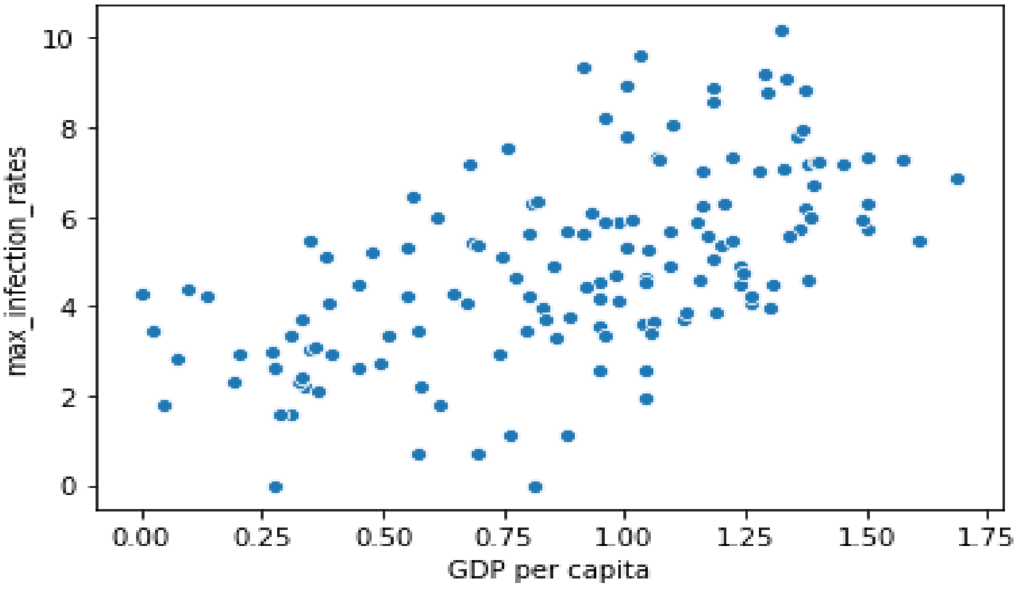
```
x = data["GDP per capita"]
y = data["max_infection_rates"]
sns.scatterplot(x,y)

#We can see the values need different scaling
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4cf22828>
```



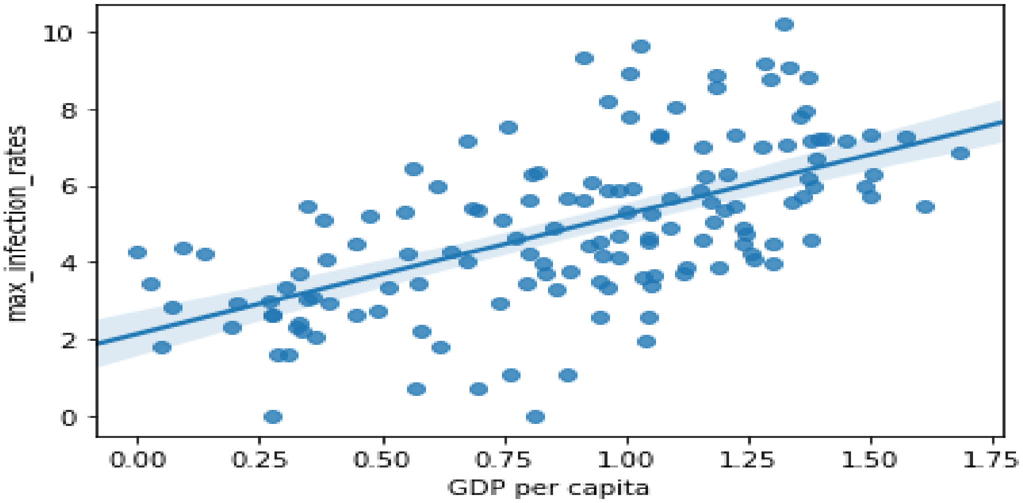
```
#Will apply log scaling to y
x = data["GDP per capita"]
y = data["max_infection_rates"]
sns.scatterplot(x,np.log(y))

#Now we can see +ve correlation
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4cf8fd30>
```



```
#RegPlot
x = data["GDP per capita"]
y = data["max_infection_rates"]

sns.regplot(x,np.log(y))

#Line fitted, +ve slope seen
```

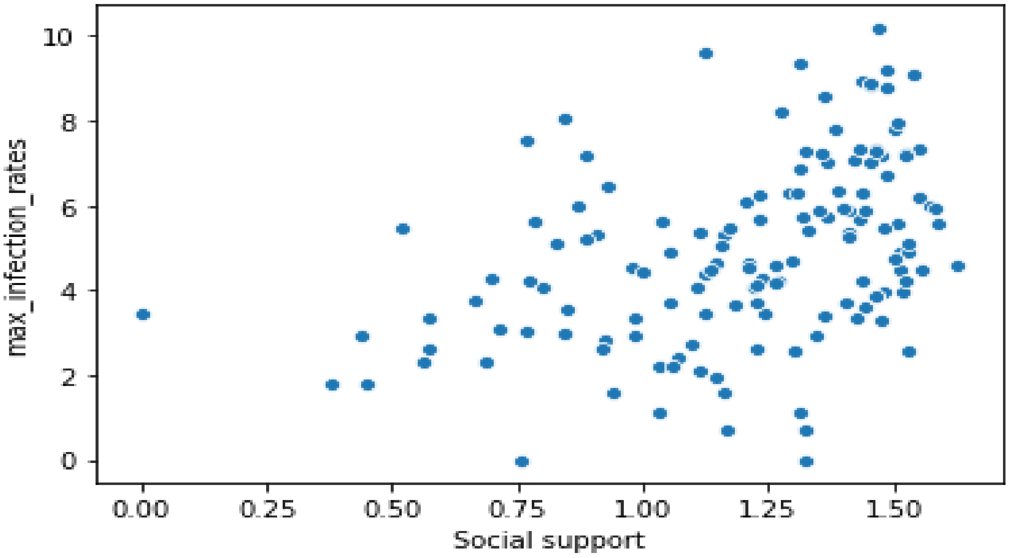
```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4cffcdd8>
```



Task 5.2: Plotting Social support vs maximum Infection rate

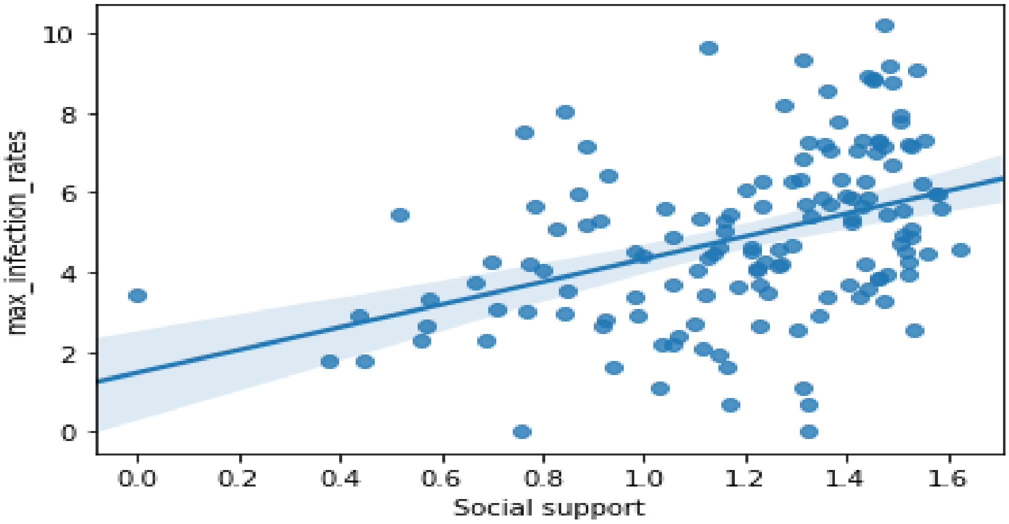
```
x = data["Social support"]
y = data["max_infection_rates"]

sns.scatterplot(x,np.log(y))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d0689b0>
```



```
x = data["Social support"]
y = data["max_infection_rates"]

sns.regplot(x,np.log(y))
```

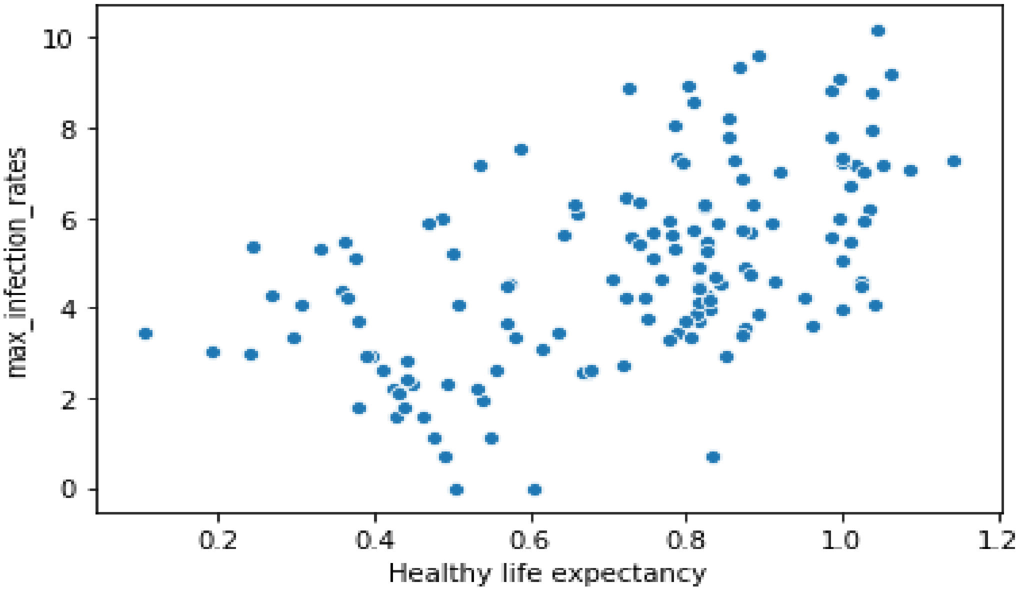
```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d0c70b8>
```



Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

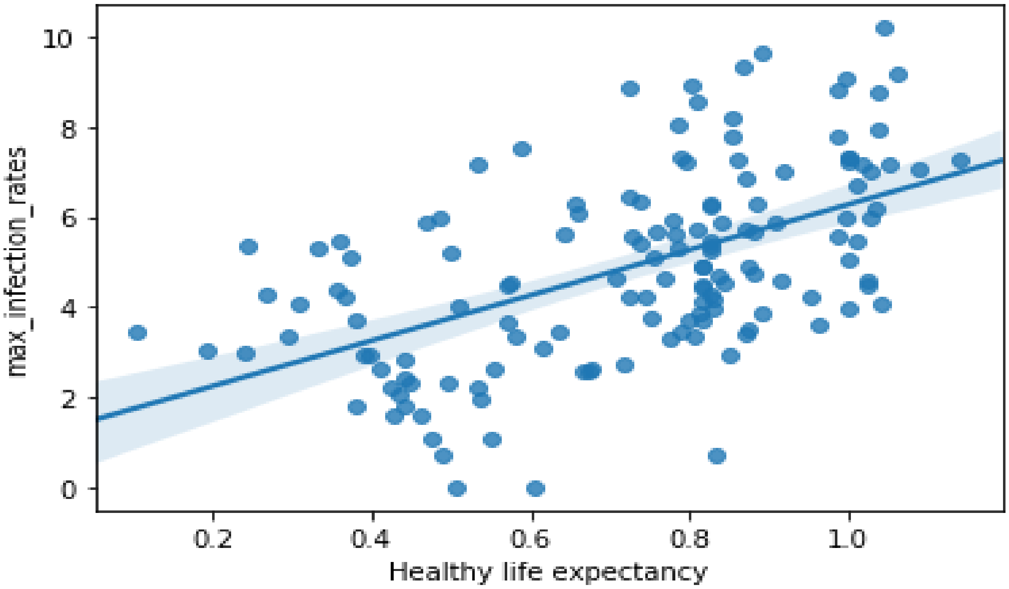
```
x = data["Healthy life expectancy"]
y = data["max_infection_rates"]

sns.scatterplot(x,np.log(y))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d1397b8>
```



```
x = data["Healthy life expectancy"]
y = data["max_infection_rates"]

sns.regplot(x,np.log(y))
```

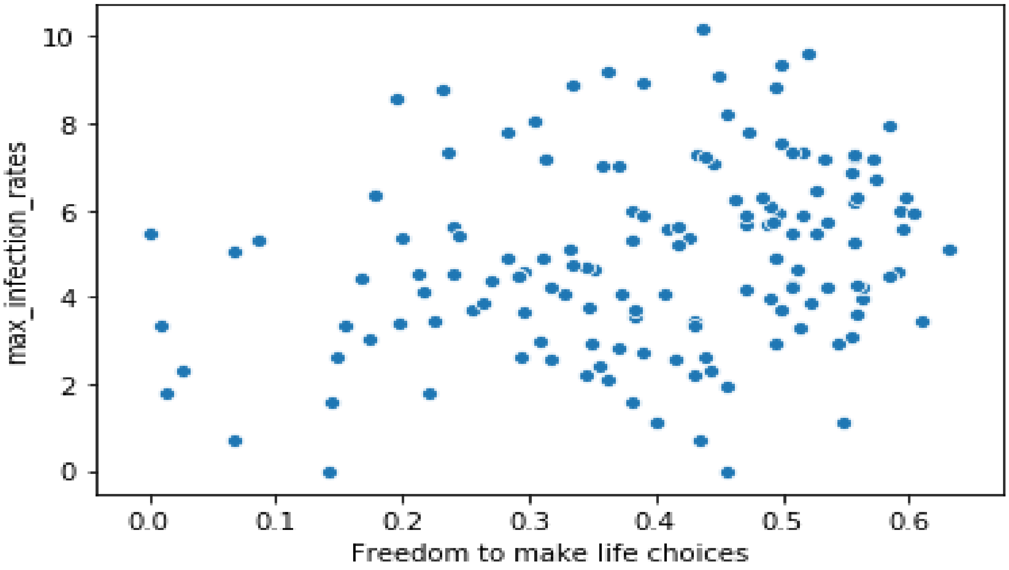
```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d1829e8>
```



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

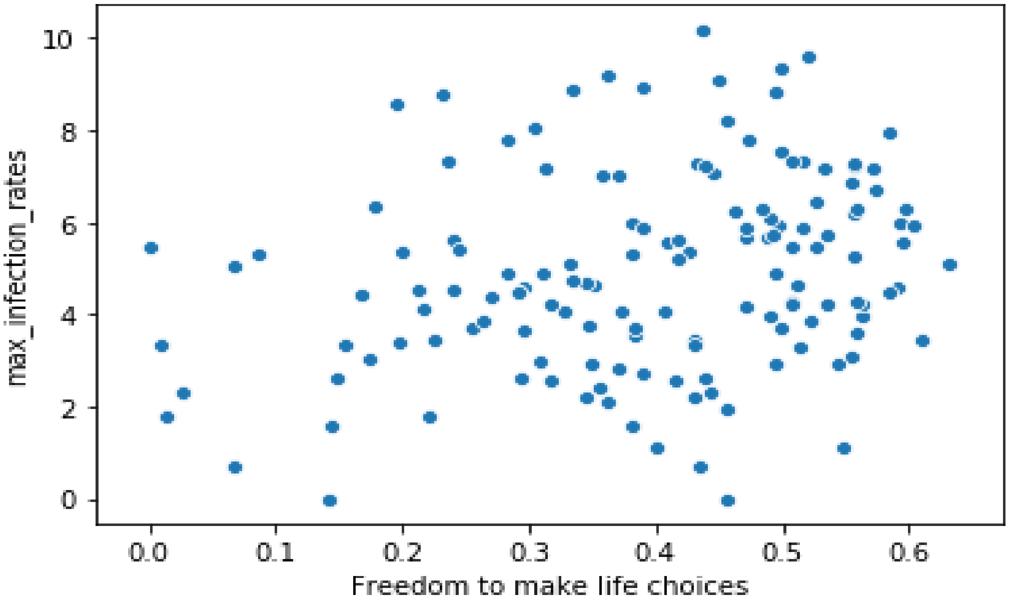
```
x = data["Freedom to make life choices"]
y = data["max_infection_rates"]

sns.scatterplot(x,np.log(y))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d1e0c88>
```



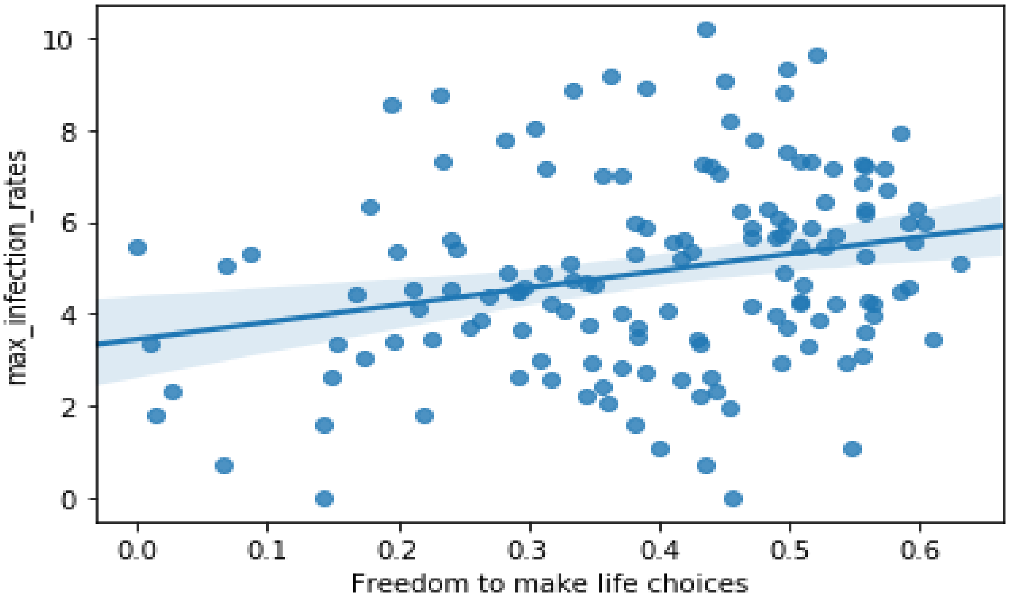
```
x = data["Freedom to make life choices"]
y = data["max_infection_rates"]

sns.regplot(x,np.log(y))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d1e0c88>
```



```
x = data["Freedom to make life choices"]
y = data["max_infection_rates"]

sns.regplot(x,np.log(y))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d4d246940>
```


max_infection_rates
Freedom to make life choices