***** The solution to the theoretical questions (questions 1 to 7 in the first part of the assignment) are sent in the “report” folder and is named “9531307_TinaGholami_Solutions of theory questions”.

A)

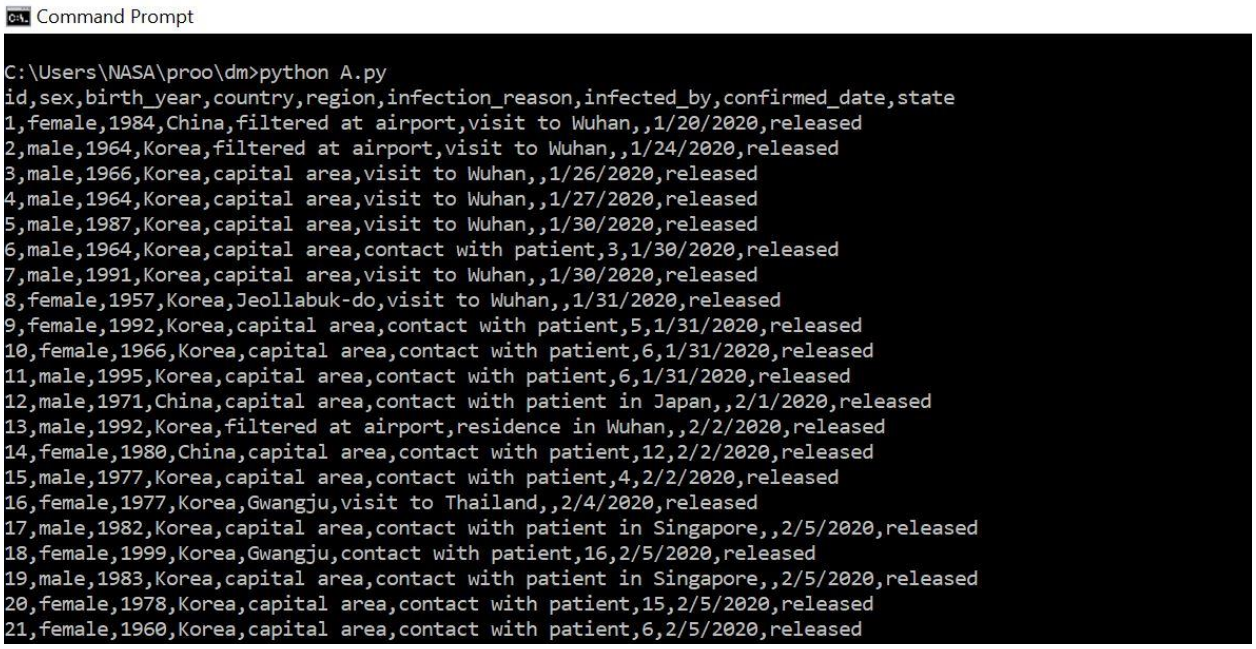
(name of the files: A.py and A2.py)

I read the file and showed it by two different methods: one in Python with no other libraries (A2.py) and two, with Pandas library (A.py). But The second method is easier and nicer when displayed.

1st method (A2.py):

```python
import csv

with open ("data.csv", newline = '') as csvfile:
    spamreader = csv.reader(csvfile, delimiter = ' ', quotechar = '|')
    for row in spamreader:
        print(' '.join(row))
```

And the output:



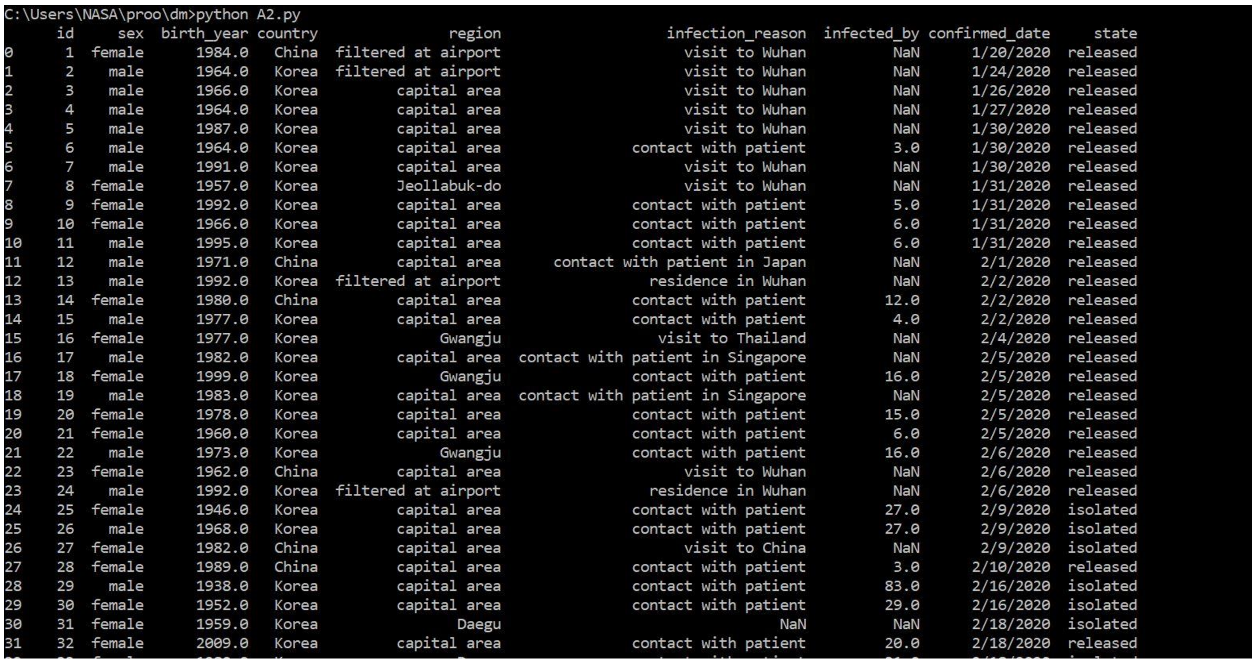
Command Prompt

```
C:\Users\NASA\proo\dm>python A.py
id,sex,birth_year,country,region,infection_reason,infected_by,confirmed_date,state
1,female,1984,China,filtered at airport,visit to Wuhan,,1/20/2020,released
2,male,1964,Korea,filtered at airport,visit to Wuhan,,1/24/2020,released
3,male,1966,Korea,capital area,visit to Wuhan,,1/26/2020,released
4,male,1964,Korea,capital area,visit to Wuhan,,1/27/2020,released
5,male,1987,Korea,capital area,visit to Wuhan,,1/30/2020,released
6,male,1964,Korea,capital area,contact with patient,3,1/30/2020,released
7,male,1991,Korea,capital area,visit to Wuhan,,1/30/2020,released
8,female,1957,Korea,Jeollabuk-do,visit to Wuhan,,1/31/2020,released
9,female,1992,Korea,capital area,contact with patient,5,1/31/2020,released
10,female,1966,Korea,capital area,contact with patient,6,1/31/2020,released
11,male,1995,Korea,capital area,contact with patient,6,1/31/2020,released
12,male,1971,China,capital area,contact with patient in Japan,,2/1/2020,released
13,male,1992,Korea,filtered at airport,residence in Wuhan,,2/2/2020,released
14,female,1980,China,capital area,contact with patient,12,2/2/2020,released
15,male,1977,Korea,capital area,contact with patient,4,2/2/2020,released
16,female,1977,Korea,Gwangju,visit to Thailand,,2/4/2020,released
17,male,1982,Korea,capital area,contact with patient in Singapore,,2/5/2020,released
18,female,1999,Korea,Gwangju,contact with patient,16,2/5/2020,released
19,male,1983,Korea,capital area,contact with patient in Singapore,,2/5/2020,released
20,female,1978,Korea,capital area,contact with patient,15,2/5/2020,released
21,female,1960,Korea,capital area,contact with patient,6,2/5/2020,released
```

Second method (A.py):

```python
import pandas as pd

df = pd.read_csv("data.csv") #data frame #read the .csv file

with pd.option_context('display.max_rows', None, 'display.max_columns', None, 'di
splay.width', None): #method number 3 (please enlarge your command window to see
the whole dataframe in allignment)
    print(df)
```

And the output:



```
C:\Users\NASA\proo\dm>python A2.py
    id     sex  birth_year country               region                  infection_reason  infected_by confirmed_date     state
0    1  female      1984.0   China  filtered at airport                    visit to Wuhan          NaN      1/20/2020  released
1    2    male      1964.0   Korea  filtered at airport                    visit to Wuhan          NaN      1/24/2020  released
2    3    male      1966.0   Korea         capital area                    visit to Wuhan          NaN      1/26/2020  released
3    4    male      1964.0   Korea         capital area                    visit to Wuhan          NaN      1/27/2020  released
4    5    male      1987.0   Korea         capital area                    visit to Wuhan          NaN      1/30/2020  released
5    6    male      1964.0   Korea         capital area              contact with patient          3.0      1/30/2020  released
6    7    male      1991.0   Korea         capital area                    visit to Wuhan          NaN      1/30/2020  released
7    8  female      1957.0   Korea         Jeollabuk-do                    visit to Wuhan          NaN      1/31/2020  released
8    9  female      1992.0   Korea         capital area              contact with patient          5.0      1/31/2020  released
9   10  female      1966.0   Korea         capital area              contact with patient          6.0      1/31/2020  released
10  11    male      1995.0   Korea         capital area              contact with patient          6.0      1/31/2020  released
11  12    male      1971.0   China         capital area     contact with patient in Japan          NaN       2/1/2020  released
12  13    male      1992.0   Korea  filtered at airport                residence in Wuhan          NaN       2/2/2020  released
13  14  female      1980.0   China         capital area              contact with patient         12.0       2/2/2020  released
14  15    male      1977.0   Korea         capital area              contact with patient          4.0       2/2/2020  released
15  16  female      1977.0   Korea              Gwangju                 visit to Thailand          NaN       2/4/2020  released
16  17    male      1982.0   Korea         capital area contact with patient in Singapore          NaN       2/5/2020  released
17  18  female      1999.0   Korea              Gwangju              contact with patient         16.0       2/5/2020  released
18  19    male      1983.0   Korea         capital area contact with patient in Singapore          NaN       2/5/2020  released
19  20  female      1978.0   Korea         capital area              contact with patient         15.0       2/5/2020  released
20  21  female      1960.0   Korea         capital area              contact with patient          6.0       2/5/2020  released
21  22    male      1973.0   Korea              Gwangju              contact with patient         16.0       2/6/2020  released
22  23  female      1962.0   China         capital area                    visit to Wuhan          NaN       2/6/2020  released
23  24    male      1992.0   Korea  filtered at airport                residence in Wuhan          NaN       2/6/2020  released
24  25  female      1946.0   Korea         capital area              contact with patient         27.0       2/9/2020  isolated
25  26    male      1968.0   Korea         capital area              contact with patient         27.0       2/9/2020  isolated
26  27  female      1982.0   China         capital area                    visit to China          NaN       2/9/2020  isolated
27  28  female      1989.0   China         capital area              contact with patient          3.0      2/10/2020  released
28  29    male      1938.0   Korea         capital area              contact with patient         83.0      2/16/2020  isolated
29  30  female      1952.0   Korea         capital area              contact with patient         29.0      2/16/2020  isolated
30  31  female      1959.0   Korea                Daegu                               NaN          NaN      2/18/2020  isolated
31  32  female      2009.0   Korea         capital area              contact with patient         20.0      2/18/2020  released
```

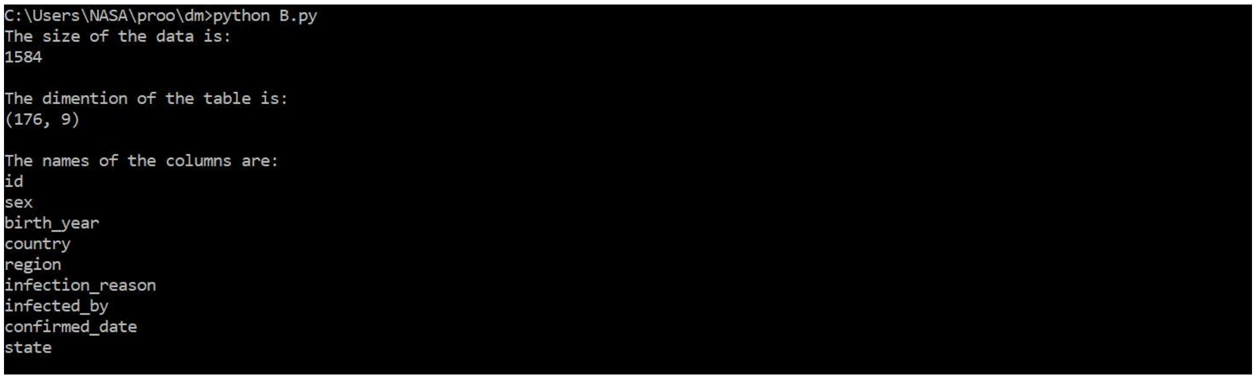
So by now, I have read the csv (comma separated values) file and showed it in my command prompt as a table. The second method is the one I chose for the next parts of the question.

B)

(The name of the file: B.py)

```
import pandas as pd

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

print("The size of the data is:")
print(df.size)

print("\nThe dimention of the table is:")
print(df.shape)

print ("\nThe names of the columns are:") #printing the names of the columns
for col in df.columns:
    print(col)
```

Ant the output:



```
C:\Users\NASA\proo\dm>python B.py
The size of the data is:
1584

The dimention of the table is:
(176, 9)

The names of the columns are:
id
sex
birth_year
country
region
infection_reason
infected_by
confirmed_date
state
```

The data is separated into 9 columns and 176 rows. The data is of all strings, integers and float.

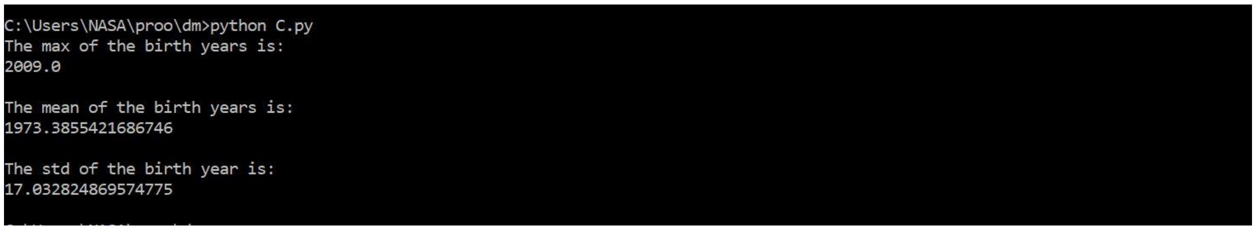
C)

(The name of the file: C.py)

I used Pandas in-built methods to calculate mean, max and standard deviation of birth_year column of the data.

```
import pandas as pd

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

print("The max of the birth years is:")
print(df['birth_year'].max())

print("\nThe mean of the birth years is:")
print(df['birth_year'].mean())

print("\nThe std of the birth year is:")
print(df['birth_year'].std())
```

And the output:

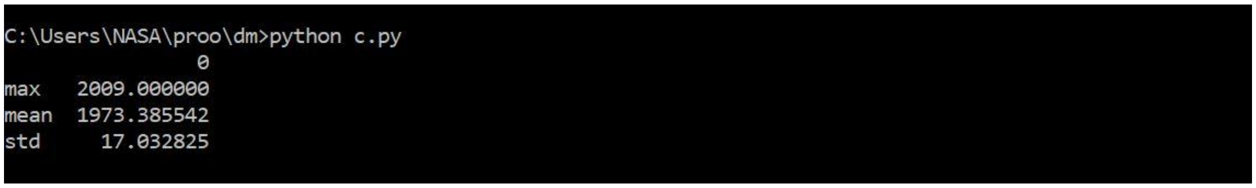


```
C:\Users\NASA\proo\dm>python C.py
The max of the birth years is:
2009.0

The mean of the birth years is:
1973.3855421686746

The std of the birth year is:
17.032824869574775
```

And if we want the output to be shown in a table, we should:

```
import pandas as pd

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

output = pd.DataFrame([df['birth_year'].max(), df['birth_year'].mean(), df['birth
_year'].std()], ['max', 'mean', 'std']) # To show output in a table
print(output)
```

And the output:



```
C:\Users\NASA\proo\dm>python c.py
                0
max   2009.000000
mean  1973.385542
std     17.032825
```

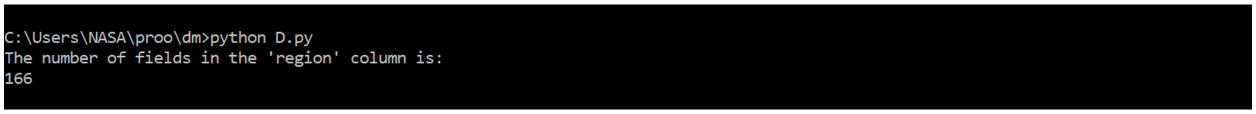
D)

(The name of the file: D.py)

I chose the “region” column:

```python
import pandas as pd

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file


print("The number of fields in the 'region' column is:")
print(df['region'].count())
```

And the output:



```
C:\Users\NASA\proo\dm>python D.py
The number of fields in the 'region' column is:
166
```

There are 166 fields in the “region” column.

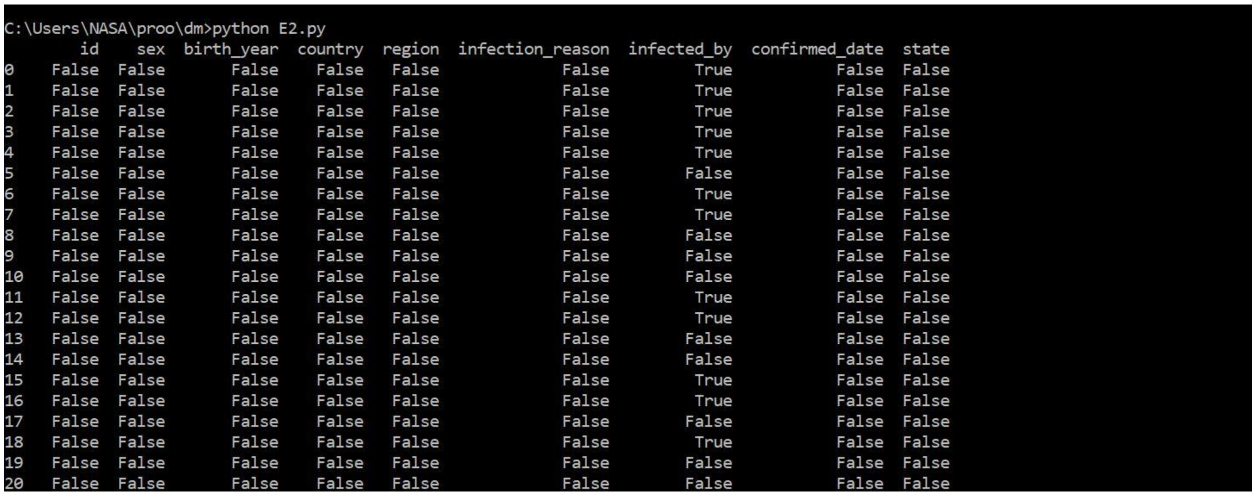
E)

(The name of the files: E.py and E2.py)

I checked the null values in data by “isna” method, which outputs a table with “False” or “True”, based on the availability of Null values. And we see in the output that there ARE null values as “NaN”s in the data frame. Then, I checked 10 solutions to handle the null values in file “E2.py”, but eventually, I chose “method 6”: interpolation and “method 10”: deleting specific rows in file “E.py”. The “E2.py” code, involves all the methods I tried to handle these null values and the “E.py” is the final solutions I chose. But here, for the sake of being concise, I only put the “E.py” code and its output.

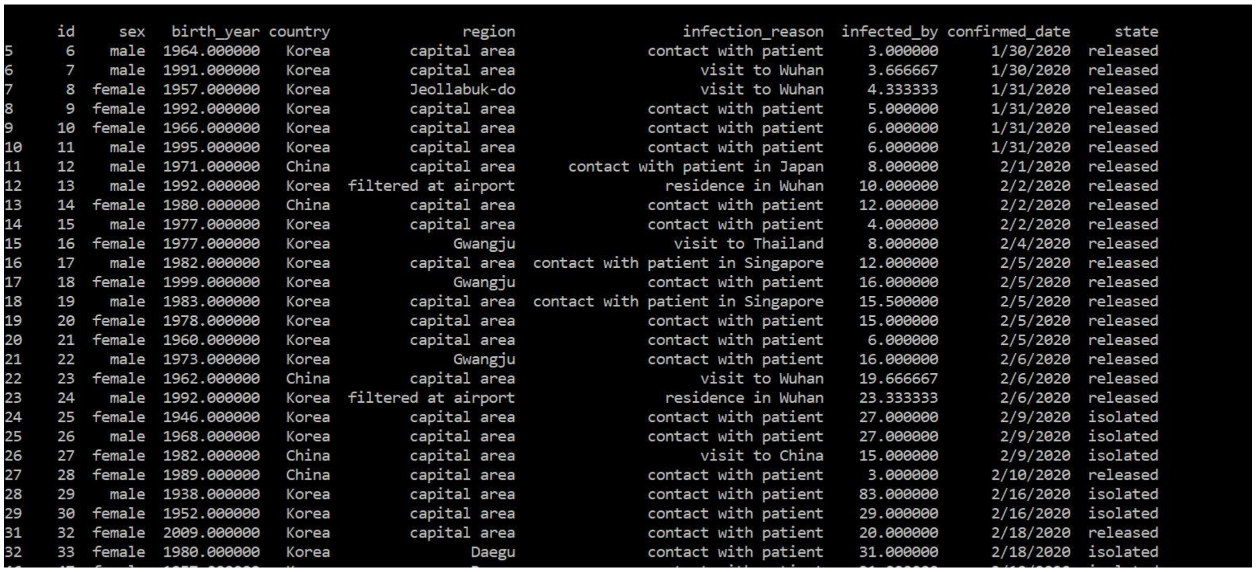
```
import pandas as pd

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file
with pd.option_context('display.max_rows', None, 'display.max_columns', None, 'di
splay.width', None):
    print(pd.isna(df))

print("\n")
#First, I use interpolate method to estimate numerical missing values, then I use
 dropna method to delete string missing values.
df_interp = df.interpolate(method = 'linear', limit_direction = 'forward') #The i
nterpolate method

df_drop = df_interp.dropna(axis = 0, how = 'any') #The dropna method
with pd.option_context('display.max_rows', None, 'display.max_columns', None, 'di
splay.width', None): #Droopping the rows that contain NaN
    print(df_drop)

print("\nOld data frame length:", len(df))
print("New data frame length:", len(df_drop))
print("Number of rows with at least 1 NaN value: ", (len(df) - len(df_drop)))
```

And the outputs:



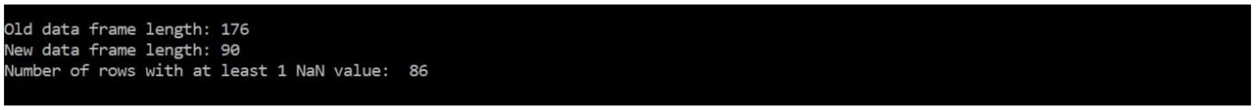
```
C:\Users\NASA\proo\dm>python E2.py
       id    sex  birth_year  country  region  infection_reason  infected_by  confirmed_date  state
0   False  False       False    False   False             False         True           False  False
1   False  False       False    False   False             False         True           False  False
2   False  False       False    False   False             False         True           False  False
3   False  False       False    False   False             False         True           False  False
4   False  False       False    False   False             False         True           False  False
5   False  False       False    False   False             False        False           False  False
6   False  False       False    False   False             False         True           False  False
7   False  False       False    False   False             False         True           False  False
8   False  False       False    False   False             False        False           False  False
9   False  False       False    False   False             False        False           False  False
10  False  False       False    False   False             False        False           False  False
11  False  False       False    False   False             False         True           False  False
12  False  False       False    False   False             False         True           False  False
13  False  False       False    False   False             False        False           False  False
14  False  False       False    False   False             False        False           False  False
15  False  False       False    False   False             False         True           False  False
16  False  False       False    False   False             False         True           False  False
17  False  False       False    False   False             False        False           False  False
18  False  False       False    False   False             False         True           False  False
19  False  False       False    False   False             False        False           False  False
20  False  False       False    False   False             False        False           False  False
```

"Isna" Method to detect whether there are Null values in the dataset, and the "True"s indicate that there are.



```
     id     sex   birth_year country               region                  infection_reason  infected_by confirmed_date     state
5     6    male  1964.000000   Korea         capital area              contact with patient     3.000000      1/30/2020  released
6     7    male  1991.000000   Korea         capital area                   visit to Wuhan     3.666667      1/30/2020  released
7     8  female  1957.000000   Korea         Jeollabuk-do                   visit to Wuhan     4.333333      1/31/2020  released
8     9  female  1992.000000   Korea         capital area              contact with patient     5.000000      1/31/2020  released
9    10  female  1966.000000   Korea         capital area              contact with patient     6.000000      1/31/2020  released
10   11    male  1995.000000   Korea         capital area              contact with patient     6.000000      1/31/2020  released
11   12    male  1971.000000   China         capital area     contact with patient in Japan     8.000000       2/1/2020  released
12   13    male  1992.000000   Korea  filtered at airport               residence in Wuhan    10.000000       2/2/2020  released
13   14  female  1980.000000   China         capital area              contact with patient    12.000000       2/2/2020  released
14   15    male  1977.000000   Korea         capital area              contact with patient     4.000000       2/2/2020  released
15   16  female  1977.000000   Korea              Gwangju                visit to Thailand     8.000000       2/4/2020  released
16   17    male  1982.000000   Korea         capital area  contact with patient in Singapore    12.000000       2/5/2020  released
17   18  female  1999.000000   Korea              Gwangju              contact with patient    16.000000       2/5/2020  released
18   19    male  1983.000000   Korea         capital area  contact with patient in Singapore    15.500000       2/5/2020  released
19   20  female  1978.000000   Korea         capital area              contact with patient    15.000000       2/5/2020  released
20   21  female  1960.000000   Korea         capital area              contact with patient     6.000000       2/5/2020  released
21   22    male  1973.000000   Korea              Gwangju              contact with patient    16.000000       2/6/2020  released
22   23  female  1962.000000   China         capital area                   visit to Wuhan    19.666667       2/6/2020  released
23   24    male  1992.000000   Korea  filtered at airport               residence in Wuhan    23.333333       2/6/2020  released
24   25  female  1946.000000   Korea         capital area              contact with patient    27.000000       2/9/2020  isolated
25   26    male  1968.000000   Korea         capital area              contact with patient    27.000000       2/9/2020  isolated
26   27  female  1982.000000   China         capital area                   visit to China    15.000000       2/9/2020  isolated
27   28  female  1989.000000   China         capital area              contact with patient     3.000000      2/10/2020  released
28   29    male  1938.000000   Korea         capital area              contact with patient    83.000000      2/16/2020  isolated
29   30  female  1952.000000   Korea         capital area              contact with patient    29.000000      2/16/2020  isolated
31   32  female  2009.000000   Korea         capital area              contact with patient    20.000000      2/18/2020  released
32   33  female  1980.000000   Korea                Daegu              contact with patient    31.000000      2/18/2020  isolated
```

Filtered data, which does not contain Null values



```
Old data frame length: 176
New data frame length: 90
Number of rows with at least 1 NaN value:  86
```

Simple calculations to show how many rows have been deleted in the process

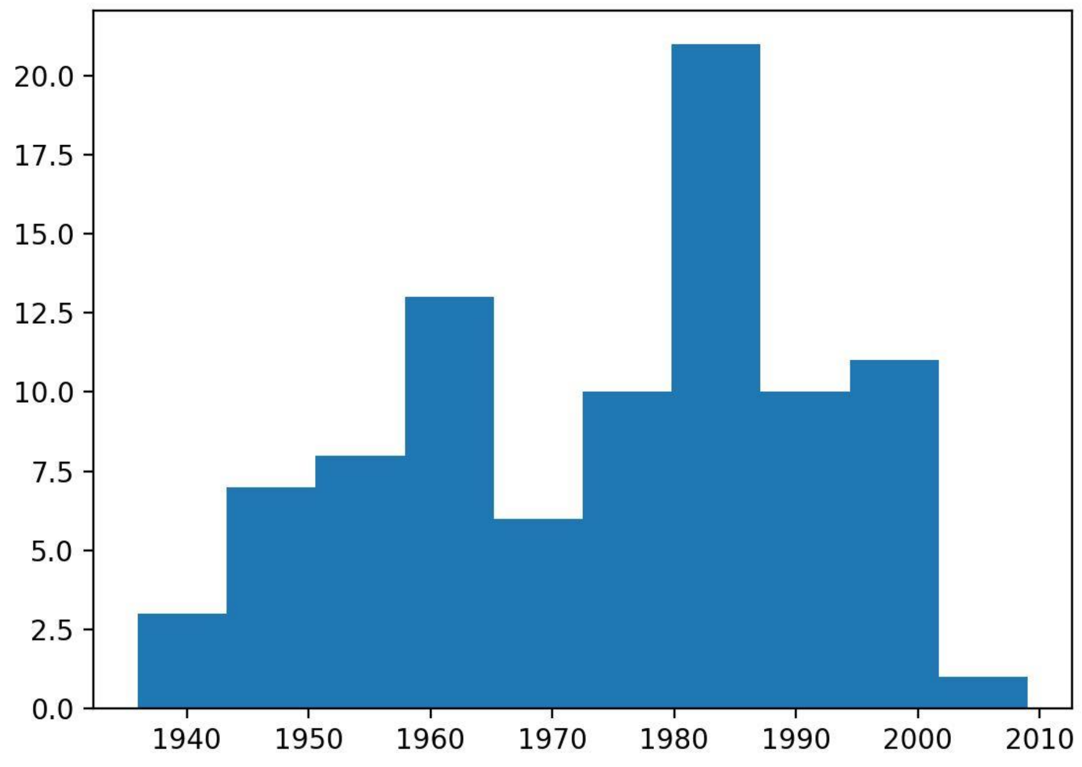
F)

(The name of the file: F.py)

I used three different visualization methods on “birth_year” column: Histogram, Boxplot, and Scatterplot.

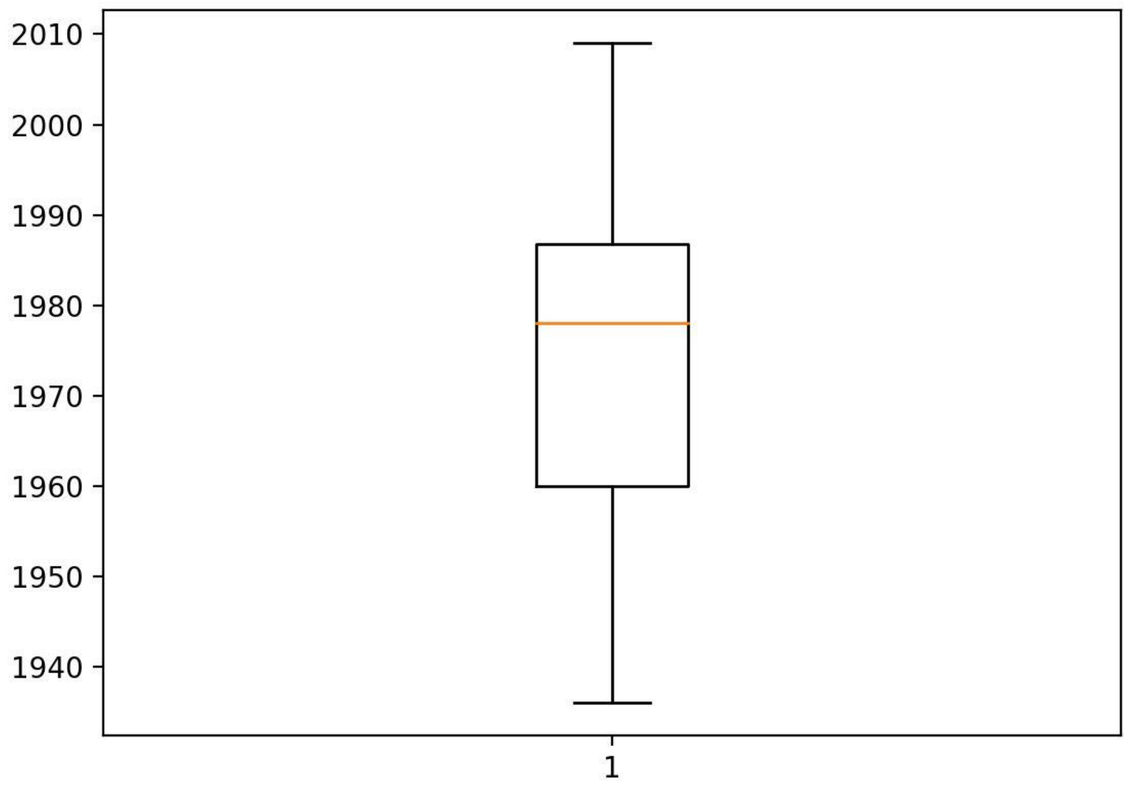
```python
import pandas as pd
from matplotlib import pyplot

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

#First, I use interpolate method to estimate numerical missing values, then I use dropna method to delete string missing values.
df_interp = df.interpolate(method = 'linear', limit_direction = 'forward') #The interpolate method
df_drop = df_interp.dropna(axis = 0, how = 'any') #The dropna method
df_new = df_drop #Therefore, our data frame from now on is "df_new".

pyplot.figure(); #Histogram Plotiing
pyplot.hist(df_new['birth_year'])
pyplot.show()
pyplot.savefig('histogram.png')

pyplot.figure(); #Boxplot Plotiing
pyplot.boxplot(df_new['birth_year'])
pyplot.show()
pyplot.savefig('boxplot.png')

pyplot.figure(); #Scatter Plotiing
pyplot.scatter(df_new['state'], df_new['birth_year'])
pyplot.show()
pyplot.savefig('scatter.png')
```

And the outputs:



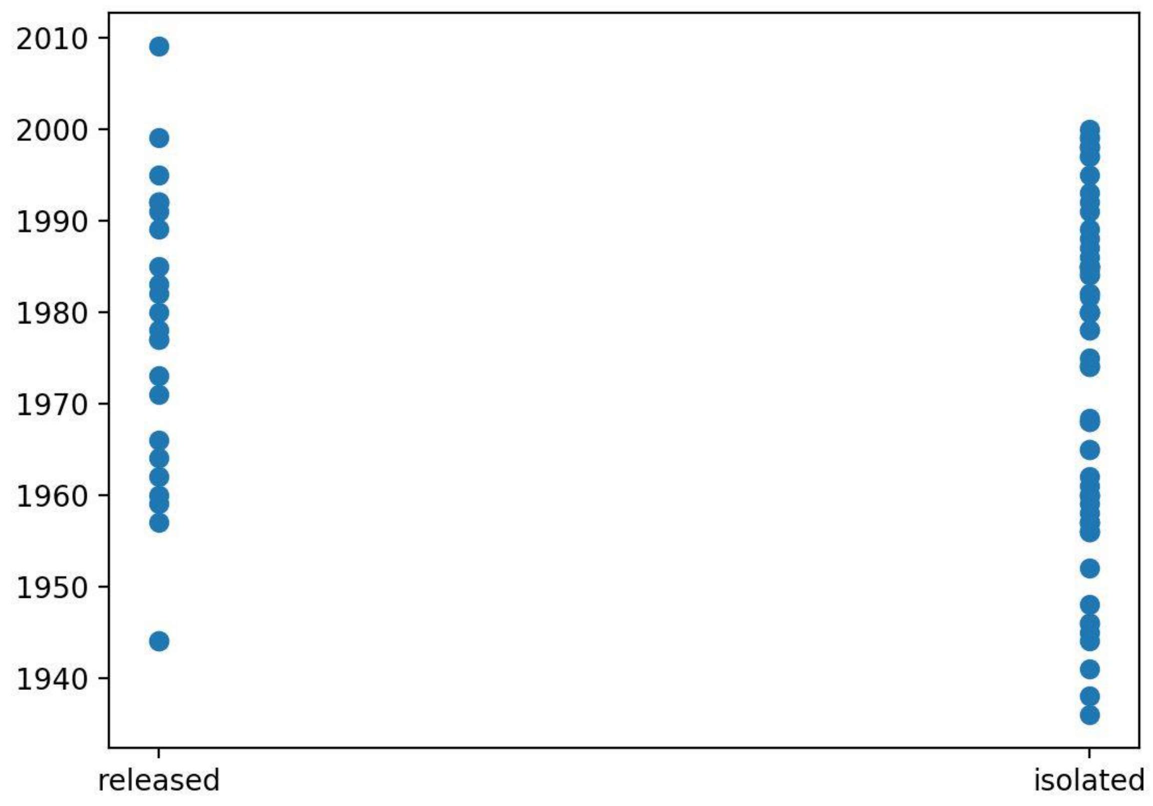
Histogram of the "birth_year" column

As we can see, the distribution of "birth_year" is the highest in the sixth bin, between the years 1980 and 1990. So most of the patients were between 40 and 50 years old.



Boxplot of the "birth_year" column

The mean of the data is in the square box, which has occurred between the years 1980 and 1970.



Scatterplot of the "birth_year" column

The scatterplot shows the data for "released" and "isolated" patients, based on the number of the incidents detected. And as we can see, the patients who were "released" are more scattered and sparse. Therefore, the probability of the existence of outlier for "released" patients is higher.

G)

(The name of the file: G.py and G2.py)

the ways to identify an outlier:

- Data point that falls outside of 3 standard deviations. we can use a z-score and if the z-score falls outside of 2 standard-deviation = the z-score method
- Data point that falls outside of 1.5 times of an interquartile range above the 3rd quartile and below the 1st quartile = IQR interquartile method
- using scatter plots

The reason for an outlier to exists in a dataset:

- Variability in the data
- An experimental measurement error

The impacts of the outlier:

Causing serious issues for statistical analysis, such as:

- Skew the data
- Significant impact on mean (**so, this shows it is better not to use mean method in detecting missing values in part (E), since we have outliers in the data and these outliers can significantly change the mean.)
- Significant impact on the standard deviation

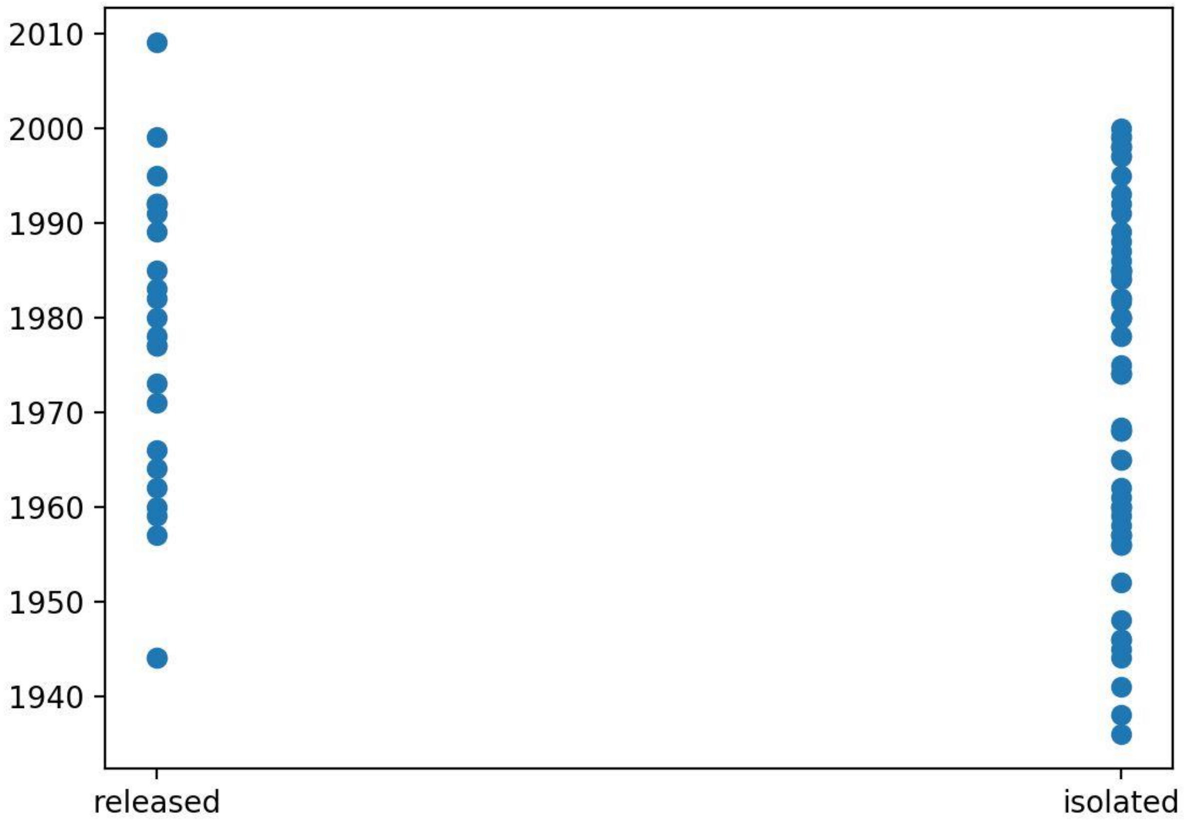
a) Z-score method (file G.py):

To check if there is outlier in the data, I showed the presence of outlier in three numerical columns “infected_by”, “birth_year”, and “id” and only the “infected_by” column contained outlier based on the “z-score” method that I used. I also used the scatterplot, but it is not precise to say that some data points are outliers based on the scatterplot since there was not very distinct data points to make one doubt of the presence of outliers and also the numerical methods such as z-score and IQR are more precise.

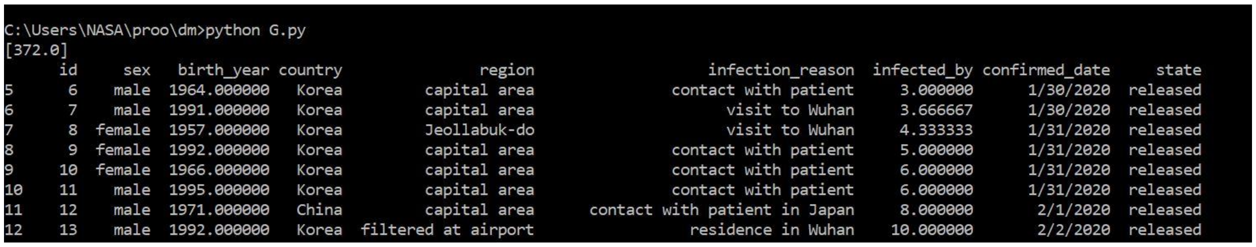
1) For the “infected_by” column

```
import pandas as pd
from matplotlib import pyplot
import numpy as np

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

#First, I use interpolate method to estimate numerical missing values, then I use
 dropna method to delete string missing values.
df_interp = df.interpolate(method = 'linear', limit_direction = 'forward') #The i
nterpolate method
df_drop = df_interp.dropna(axis = 0, how = 'any') #The dropna method
df_new = df_drop #Therefore, our data frame from now on is "df_new".
df_new_column = df_new['infected_by']
# df_new_column = df_new['birth_year']
# df_new_column = df_new['id']

#Method 1: Using scatterplot to detect outliers:
pyplot.figure() #Scatter Plotiing
pyplot.scatter(df_new['state'], df_new['birth_year'])
pyplot.show()

#Method 2: Using Z-score to detect outliers:
outliers = []

def detect_outlier(data):

    threshold = 3
    mean_data = np.mean(data)
    std_data = np.std(data)

    for item in data:
        z_score = (item - mean_data) / std_data
        if np.abs(z_score) > threshold:
            outliers.append(item)
    return outliers

outlier_df = detect_outlier(df_new_column)
print(outlier_df)

df_filtered = df_new.drop([162])
with pd.option_context('display.max_rows', None, 'display.max_columns', None, 'di
splay.width', None):
    print(df_filtered)
```
As we can see above, an outlier was detected in row number 162, so I dropped this row in order to handle the outliers.

And the output:



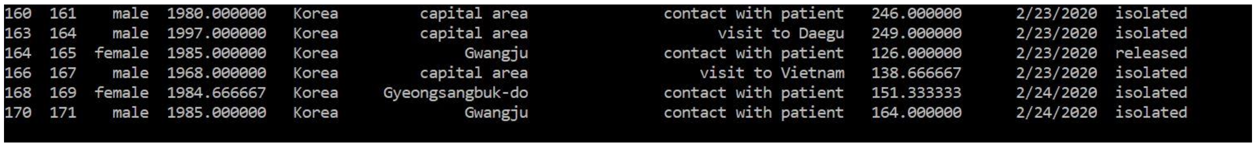
Scatterplot for the "infected_by" column to check for outliers



```
C:\Users\NASA\proo\dm>python G.py
[372.0]
     id     sex  birth_year country               region                infection_reason  infected_by confirmed_date     state
5     6    male  1964.000000   Korea         capital area            contact with patient     3.000000      1/30/2020  released
6     7    male  1991.000000   Korea         capital area                  visit to Wuhan     3.666667      1/30/2020  released
7     8  female  1957.000000   Korea         Jeollabuk-do                  visit to Wuhan     4.333333      1/31/2020  released
8     9  female  1992.000000   Korea         capital area            contact with patient     5.000000      1/31/2020  released
9    10  female  1966.000000   Korea         capital area            contact with patient     6.000000      1/31/2020  released
10   11    male  1995.000000   Korea         capital area            contact with patient     6.000000      1/31/2020  released
11   12    male  1971.000000   China         capital area   contact with patient in Japan     8.000000       2/1/2020  released
12   13    male  1992.000000   Korea  filtered at airport              residence in Wuhan    10.000000       2/2/2020  released
```

We notice that element [372.0] in the "infected_by" column was detected as an outlier.

And since this data belongs to row number 162, this row is deleted from the data frame after it was detected by the z-score method, below:



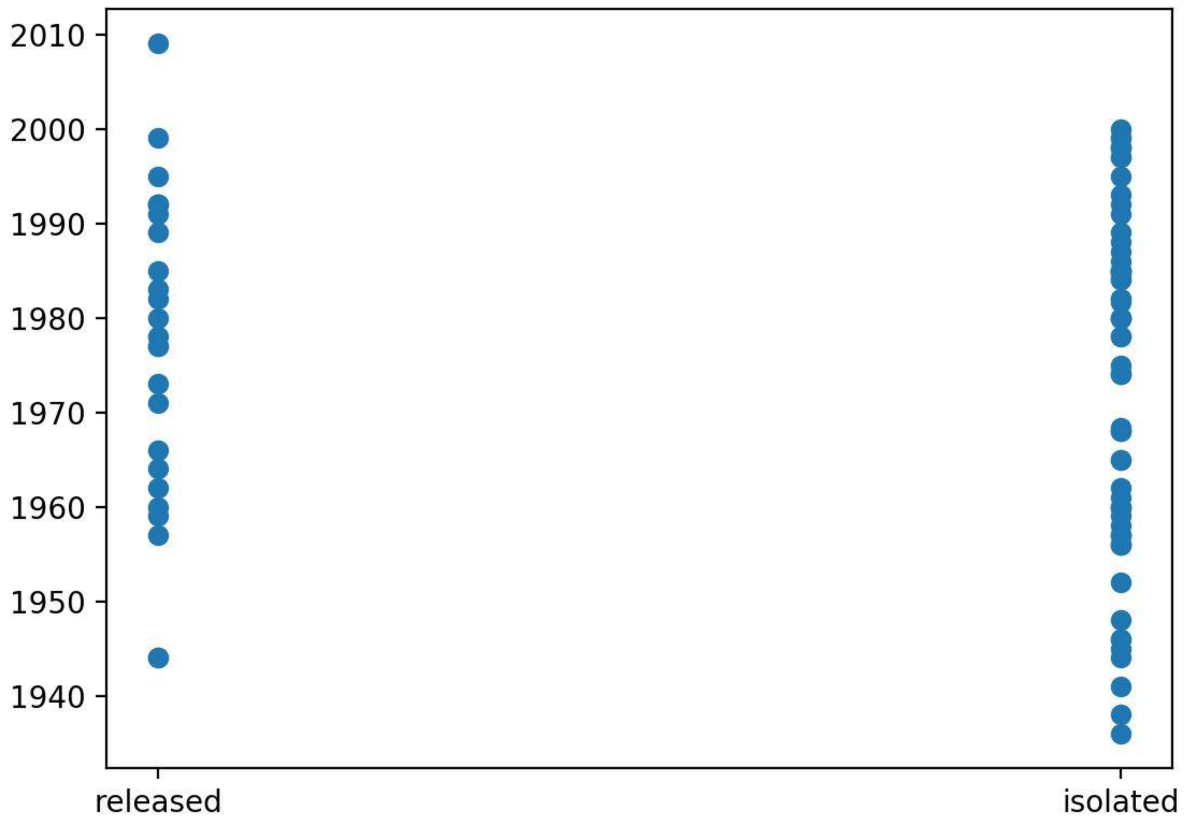
```
160  161    male  1980.000000   Korea         capital area            contact with patient   246.000000      2/23/2020  isolated
163  164    male  1997.000000   Korea         capital area                  visit to Daegu   249.000000      2/23/2020  isolated
164  165  female  1985.000000   Korea              Gwangju            contact with patient   126.000000      2/23/2020  released
166  167    male  1968.000000   Korea         capital area                visit to Vietnam   138.666667      2/23/2020  isolated
168  169  female  1984.666667   Korea     Gyeongsangbuk-do            contact with patient   151.333333      2/24/2020  isolated
170  171    male  1985.000000   Korea              Gwangju            contact with patient   164.000000      2/24/2020  isolated
```

2) For the “birth_year” column:

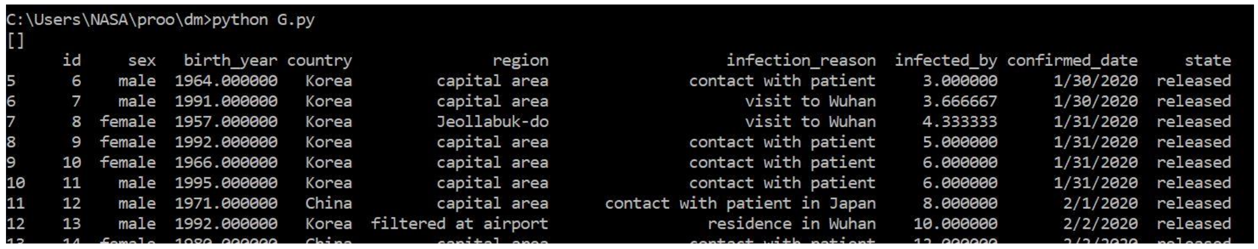
We only need to un-command this line of the code:

```
df_new_column = df_new['birth_year']
```

And the outputs are:



Scatterplot for the “birth_year” column to check for outliers



```
C:\Users\NASA\proo\dm>python G.py
[]
      id     sex   birth_year country                region                  infection_reason  infected_by confirmed_date     state
5      6    male  1964.000000   Korea          capital area              contact with patient     3.000000      1/30/2020  released
6      7    male  1991.000000   Korea          capital area                    visit to Wuhan     3.666667      1/30/2020  released
7      8  female  1957.000000   Korea           Jeollabuk-do                    visit to Wuhan     4.333333      1/31/2020  released
8      9  female  1992.000000   Korea          capital area              contact with patient     5.000000      1/31/2020  released
9     10  female  1966.000000   Korea          capital area              contact with patient     6.000000      1/31/2020  released
10    11    male  1995.000000   Korea          capital area              contact with patient     6.000000      1/31/2020  released
11    12    male  1971.000000   China          capital area     contact with patient in Japan     8.000000       2/1/2020  released
12    13    male  1992.000000   Korea    filtered at airport               residence in Wuhan    10.000000       2/2/2020  released
```

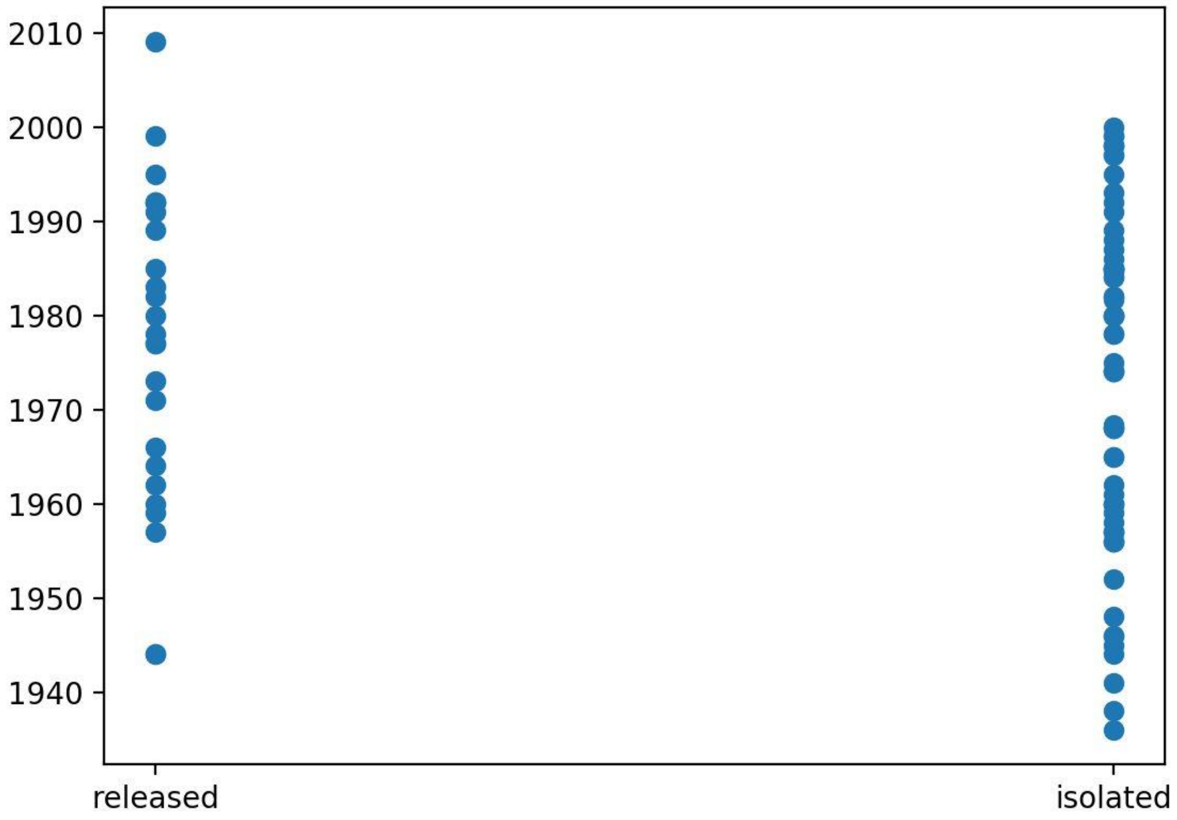
As we see, no outlier was detected, therefore no row will be dropped.

3) For the “id” column:

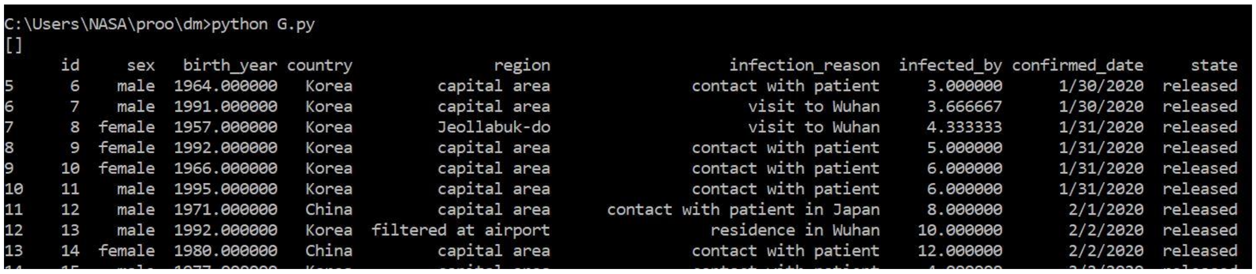
We only need to un-command this line of the code:

```
df_new_column = df_new['id']
```

And the outputs are:



Scatterplot for the “id” column to check for outliers



```
C:\Users\NASA\proo\dm>python G.py
[]
     id     sex   birth_year country               region                 infection_reason  infected_by confirmed_date     state
5     6    male  1964.000000   Korea         capital area             contact with patient     3.000000      1/30/2020  released
6     7    male  1991.000000   Korea         capital area                   visit to Wuhan     3.666667      1/30/2020  released
7     8  female  1957.000000   Korea         Jeollabuk-do                   visit to Wuhan     4.333333      1/31/2020  released
8     9  female  1992.000000   Korea         capital area             contact with patient     5.000000      1/31/2020  released
9    10  female  1966.000000   Korea         capital area             contact with patient     6.000000      1/31/2020  released
10   11    male  1995.000000   Korea         capital area             contact with patient     6.000000      1/31/2020  released
11   12    male  1971.000000   China         capital area    contact with patient in Japan     8.000000       2/1/2020  released
12   13    male  1992.000000   Korea  filtered at airport               residence in Wuhan    10.000000       2/2/2020  released
13   14  female  1980.000000   China         capital area             contact with patient    12.000000       2/2/2020  released
```

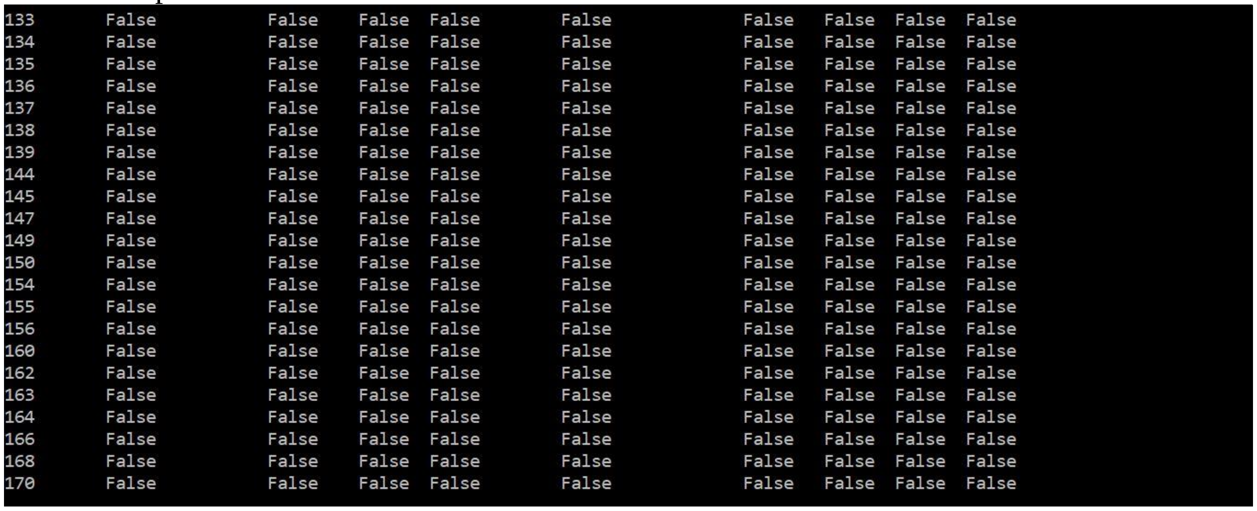
Again in here, the system does not detect any outliers in the “id” column, thus no row will be deleted.

b) IQR Interquartile method (file G2.py):

IQR tells how spread the middle values are. It can be used to tell when a value is too far from the middle. An outlier is a point which falls more than 1.5 times the interquartile range above the third quartile or below the first quartile.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("data.csv") #Reading the data frame fro a .csv file

#First, I use interpolate method to estimate numerical missing values, then I use
 dropna method to delete string missing values.
df_interp = df.interpolate(method = 'linear', limit_direction = 'forward') #The i
nterpolate method
df_drop = df_interp.dropna(axis = 0, how = 'any') #The dropna method
df_new = df_drop #Therefore, our data frame from now on is "df_new".

q1 = df_new.quantile(0.25) #Calculating the quartiles
q3 = df_new.quantile(0.75)
iqr = q3 - q1

with pd.option_context('display.max_rows', None, 'display.max_columns', None, 'di
splay.width', None): #Showing the outliers
    print((df_new <= (q1 - 1.5 * iqr)) | (df_new >= (q3 + 1.5 * iqr)))

# df_drop = df_interp.dropna([]) #Dropping the row containing outlier
```

And the output:



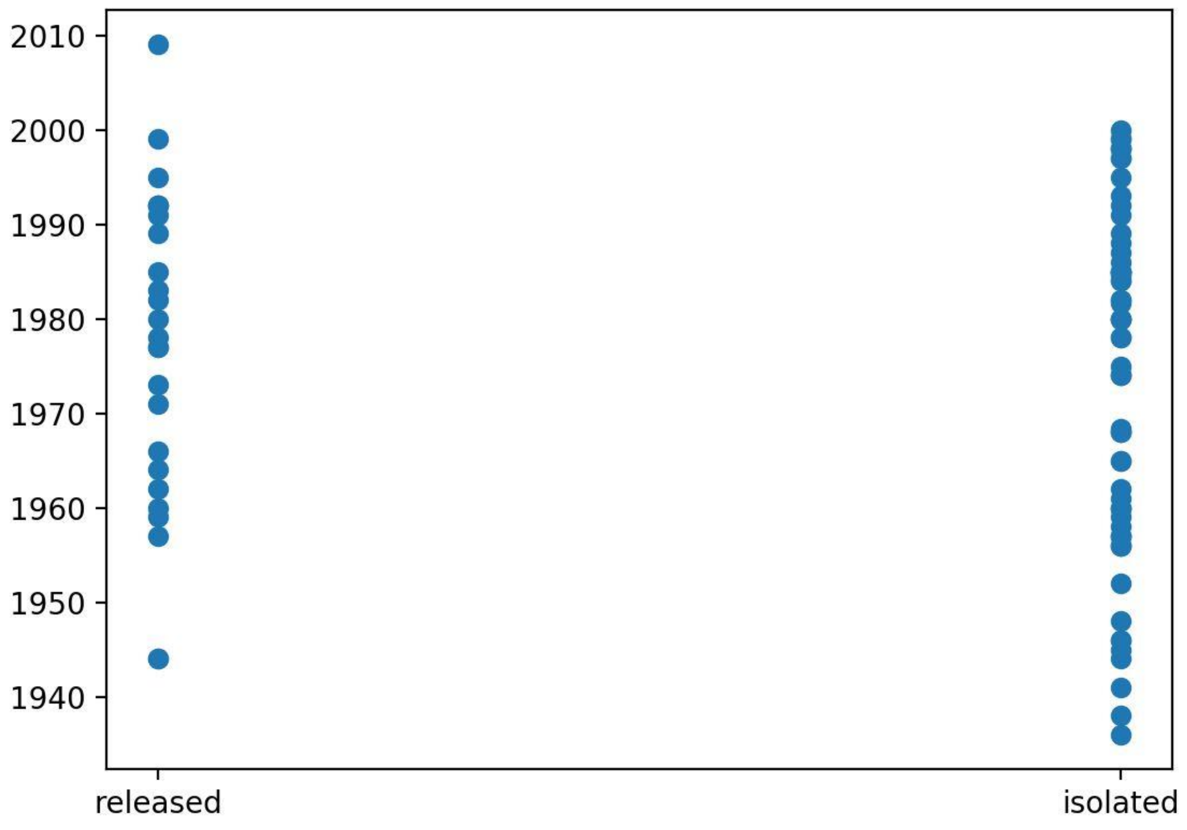
```
133       False          False    False  False        False            False  False  False  False
134       False          False    False  False        False            False  False  False  False
135       False          False    False  False        False            False  False  False  False
136       False          False    False  False        False            False  False  False  False
137       False          False    False  False        False            False  False  False  False
138       False          False    False  False        False            False  False  False  False
139       False          False    False  False        False            False  False  False  False
144       False          False    False  False        False            False  False  False  False
145       False          False    False  False        False            False  False  False  False
147       False          False    False  False        False            False  False  False  False
149       False          False    False  False        False            False  False  False  False
150       False          False    False  False        False            False  False  False  False
154       False          False    False  False        False            False  False  False  False
155       False          False    False  False        False            False  False  False  False
156       False          False    False  False        False            False  False  False  False
160       False          False    False  False        False            False  False  False  False
162       False          False    False  False        False            False  False  False  False
163       False          False    False  False        False            False  False  False  False
164       False          False    False  False        False            False  False  False  False
166       False          False    False  False        False            False  False  False  False
168       False          False    False  False        False            False  False  False  False
170       False          False    False  False        False            False  False  False  False
```

Which shows that unlike Z-score method, the IQR method didn't detect any outlier. (in the Z-score method, we had an outlier in row 162 for "infected_by" column.)

c) Scatterplot method:

Just like part (F), the scatter plot for the "birth_year" data is:

1) Scatterplot for "birth_year" column:



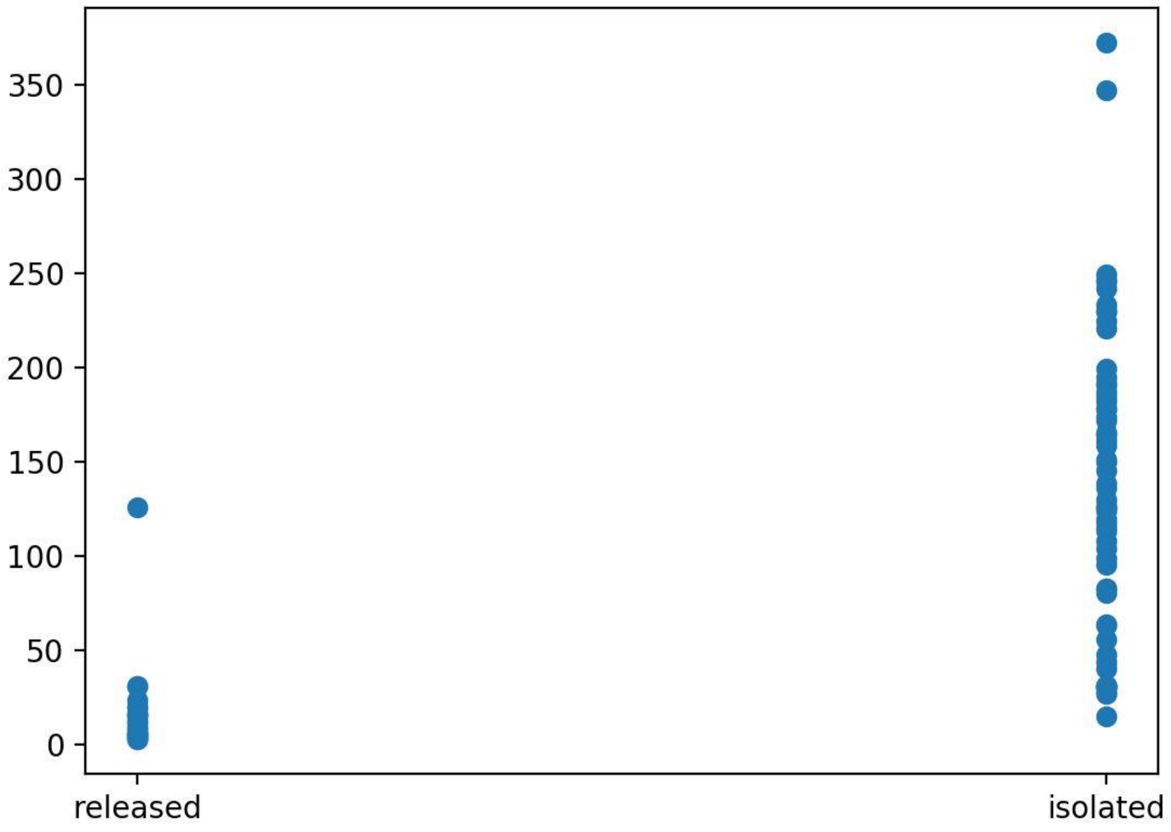
Scatterplot for the "birth_year" data

As I guess just by looking at the histogram above, there seems to be two outliers in the "released" data, one between 1940 and 1950 and the other between 2000 to 2010.

But if we take these two points as outliers, we should consider the relative data points between those two year intervals for the "isolated" data as outliers, as well.

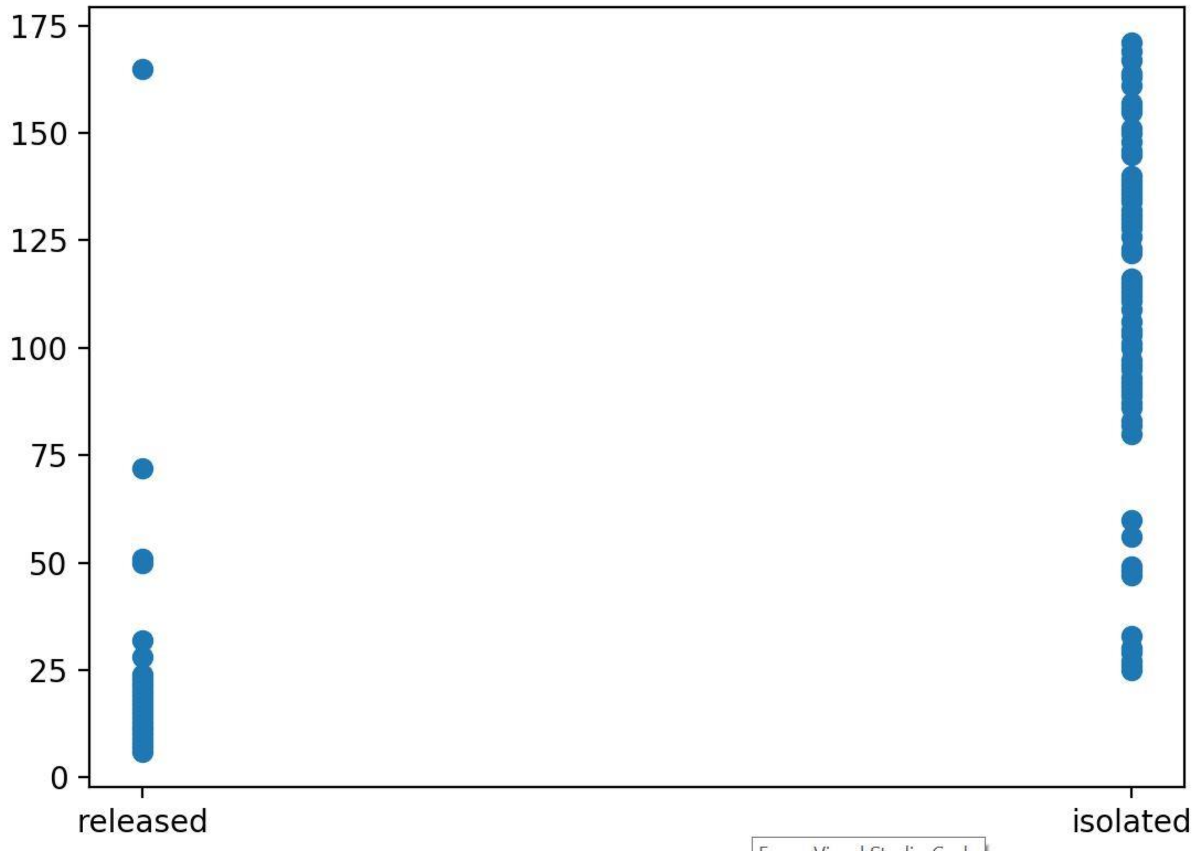
But should we really take any data points between years (1940 and 1950) and (2000 and 2010) as outliers? Well I think we cannot answer this question with certainty just by looking at the histogram above. As I have mentioned in Z-score and IQR method, we'd better employ numerical methods to detect outliers, since when the data is big, like in this scenario, we should decide which data points are outliers by measuring their relative distance from the mean of all the data point we have in hand.

2) Scatterplot for "infected_by" column:



Scatterplot for the "infected_by" data

3) Scatterplot for “id” column:



Scatterplot for the “id” data