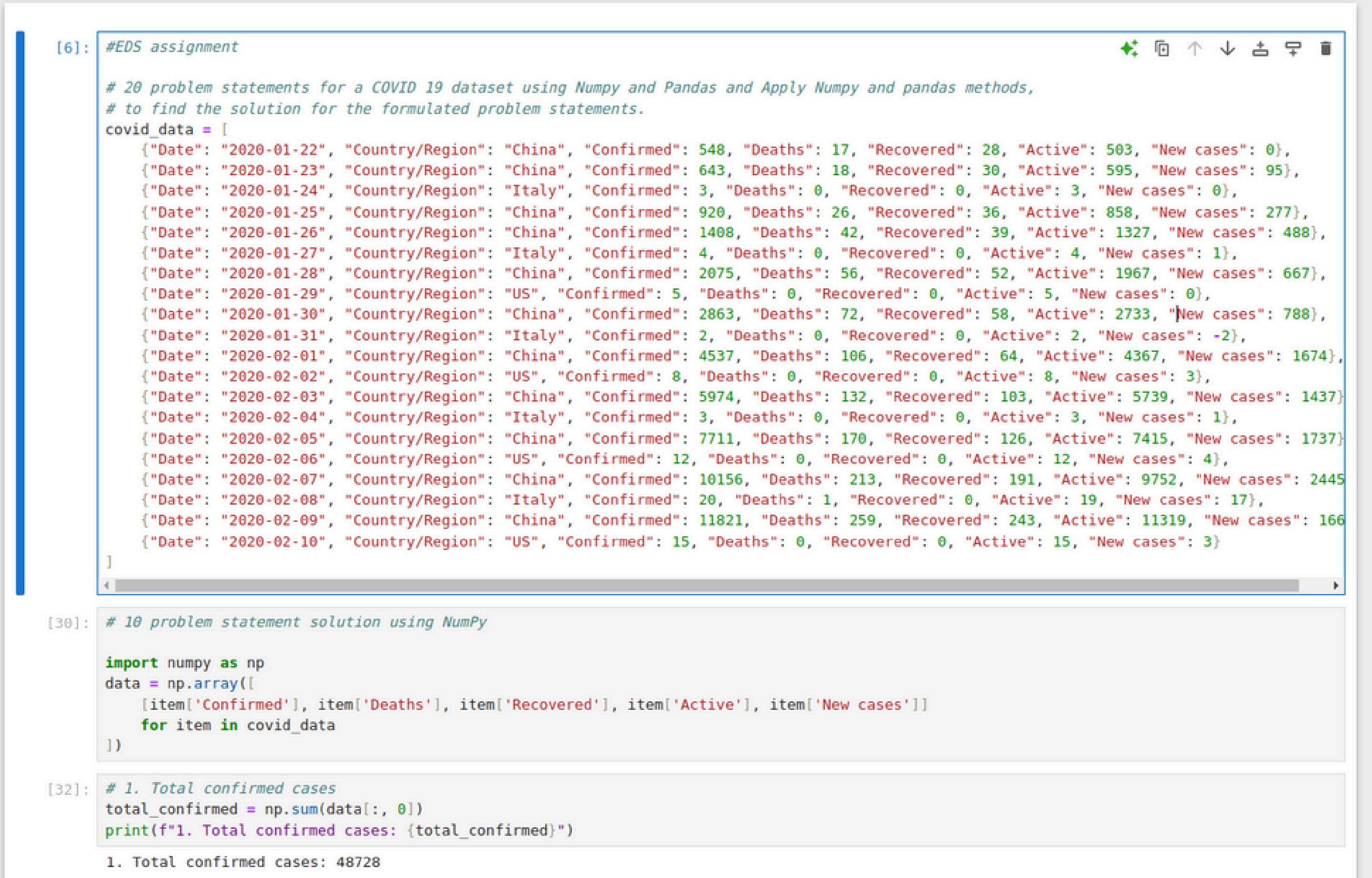
EDS Assignment

Name: Harshini Ramdas Bhandary
PRN: 202401040151
Roll no: CS2-03
Division: CS2
Batch: C21

20 problem statements for a given dataset using Numpy and Pandas and Apply Numpy and pandas methods to find the solution for the formulated problem statements.

DATASET : COVID-19

10 Problem statements and solution using NumPy



[6]:

```
#EDS assignment

# 20 problem statements for a COVID 19 dataset using Numpy and Pandas and Apply Numpy and pandas methods,
# to find the solution for the formulated problem statements.
covid_data = [
    {"Date": "2020-01-22", "Country/Region": "China", "Confirmed": 548, "Deaths": 17, "Recovered": 28, "Active": 503, "New cases": 0},
    {"Date": "2020-01-23", "Country/Region": "China", "Confirmed": 643, "Deaths": 18, "Recovered": 30, "Active": 595, "New cases": 95},
    {"Date": "2020-01-24", "Country/Region": "Italy", "Confirmed": 3, "Deaths": 0, "Recovered": 0, "Active": 3, "New cases": 0},
    {"Date": "2020-01-25", "Country/Region": "China", "Confirmed": 920, "Deaths": 26, "Recovered": 36, "Active": 858, "New cases": 277},
    {"Date": "2020-01-26", "Country/Region": "China", "Confirmed": 1408, "Deaths": 42, "Recovered": 39, "Active": 1327, "New cases": 488},
    {"Date": "2020-01-27", "Country/Region": "Italy", "Confirmed": 4, "Deaths": 0, "Recovered": 0, "Active": 4, "New cases": 1},
    {"Date": "2020-01-28", "Country/Region": "China", "Confirmed": 2075, "Deaths": 56, "Recovered": 52, "Active": 1967, "New cases": 667},
    {"Date": "2020-01-29", "Country/Region": "US", "Confirmed": 5, "Deaths": 0, "Recovered": 0, "Active": 5, "New cases": 0},
    {"Date": "2020-01-30", "Country/Region": "China", "Confirmed": 2863, "Deaths": 72, "Recovered": 58, "Active": 2733, "New cases": 788},
    {"Date": "2020-01-31", "Country/Region": "Italy", "Confirmed": 2, "Deaths": 0, "Recovered": 0, "Active": 2, "New cases": -2},
    {"Date": "2020-02-01", "Country/Region": "China", "Confirmed": 4537, "Deaths": 106, "Recovered": 64, "Active": 4367, "New cases": 1674},
    {"Date": "2020-02-02", "Country/Region": "US", "Confirmed": 8, "Deaths": 0, "Recovered": 0, "Active": 8, "New cases": 3},
    {"Date": "2020-02-03", "Country/Region": "China", "Confirmed": 5974, "Deaths": 132, "Recovered": 103, "Active": 5739, "New cases": 1437}
    {"Date": "2020-02-04", "Country/Region": "Italy", "Confirmed": 3, "Deaths": 0, "Recovered": 0, "Active": 3, "New cases": 1},
    {"Date": "2020-02-05", "Country/Region": "China", "Confirmed": 7711, "Deaths": 170, "Recovered": 126, "Active": 7415, "New cases": 1737}
    {"Date": "2020-02-06", "Country/Region": "US", "Confirmed": 12, "Deaths": 0, "Recovered": 0, "Active": 12, "New cases": 4},
    {"Date": "2020-02-07", "Country/Region": "China", "Confirmed": 10156, "Deaths": 213, "Recovered": 191, "Active": 9752, "New cases": 2445
    {"Date": "2020-02-08", "Country/Region": "Italy", "Confirmed": 20, "Deaths": 1, "Recovered": 0, "Active": 19, "New cases": 17},
    {"Date": "2020-02-09", "Country/Region": "China", "Confirmed": 11821, "Deaths": 259, "Recovered": 243, "Active": 11319, "New cases": 166
    {"Date": "2020-02-10", "Country/Region": "US", "Confirmed": 15, "Deaths": 0, "Recovered": 0, "Active": 15, "New cases": 3}
]
```

[30]:

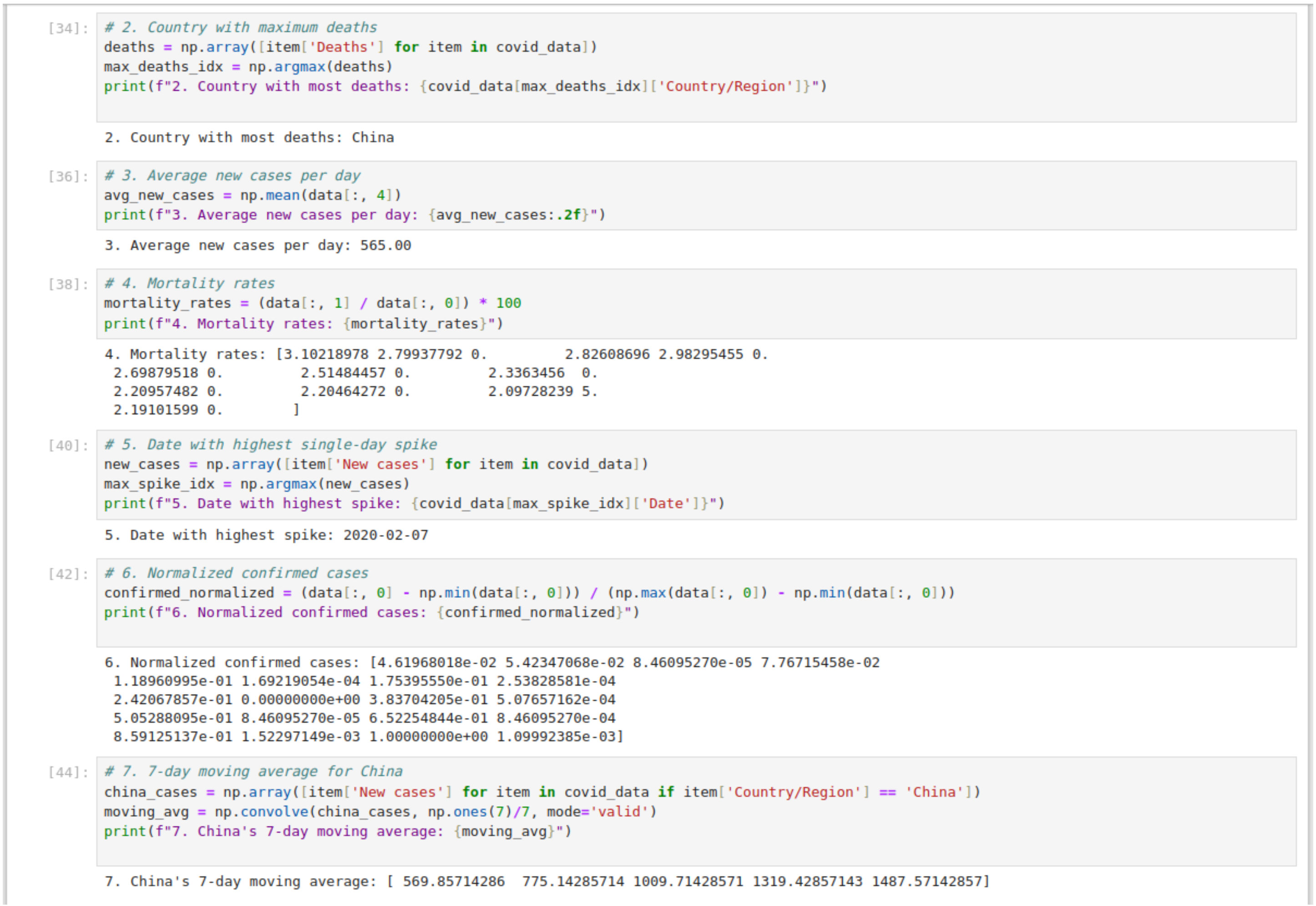
```
# 10 problem statement solution using NumPy

import numpy as np
data = np.array([
    [item['Confirmed'], item['Deaths'], item['Recovered'], item['Active'], item['New cases']]
    for item in covid_data
])
```

[32]:

```
# 1. Total confirmed cases
total_confirmed = np.sum(data[:, 0])
print(f"1. Total confirmed cases: {total_confirmed}")
```

```
1. Total confirmed cases: 48728
```



[34]:

```
# 2. Country with maximum deaths
deaths = np.array([item['Deaths'] for item in covid_data])
max_deaths_idx = np.argmax(deaths)
print(f"2. Country with most deaths: {covid_data[max_deaths_idx]['Country/Region']}")
```

```
2. Country with most deaths: China
```

[36]:

```
# 3. Average new cases per day
avg_new_cases = np.mean(data[:, 4])
print(f"3. Average new cases per day: {avg_new_cases:.2f}")
```

```
3. Average new cases per day: 565.00
```

[38]:

```
# 4. Mortality rates
mortality_rates = (data[:, 1] / data[:, 0]) * 100
print(f"4. Mortality rates: {mortality_rates}")
```

```
4. Mortality rates: [3.10218978 2.79937792 0.         2.82608696 2.98295455 0.
 2.69879518 0.         2.51484457 0.         2.3363456  0.
 2.20957482 0.         2.20464272 0.         2.09728239 5.
 2.19101599 0.        ]
```

[40]:

```
# 5. Date with highest single-day spike
new_cases = np.array([item['New cases'] for item in covid_data])
max_spike_idx = np.argmax(new_cases)
print(f"5. Date with highest spike: {covid_data[max_spike_idx]['Date']}")
```

```
5. Date with highest spike: 2020-02-07
```

[42]:

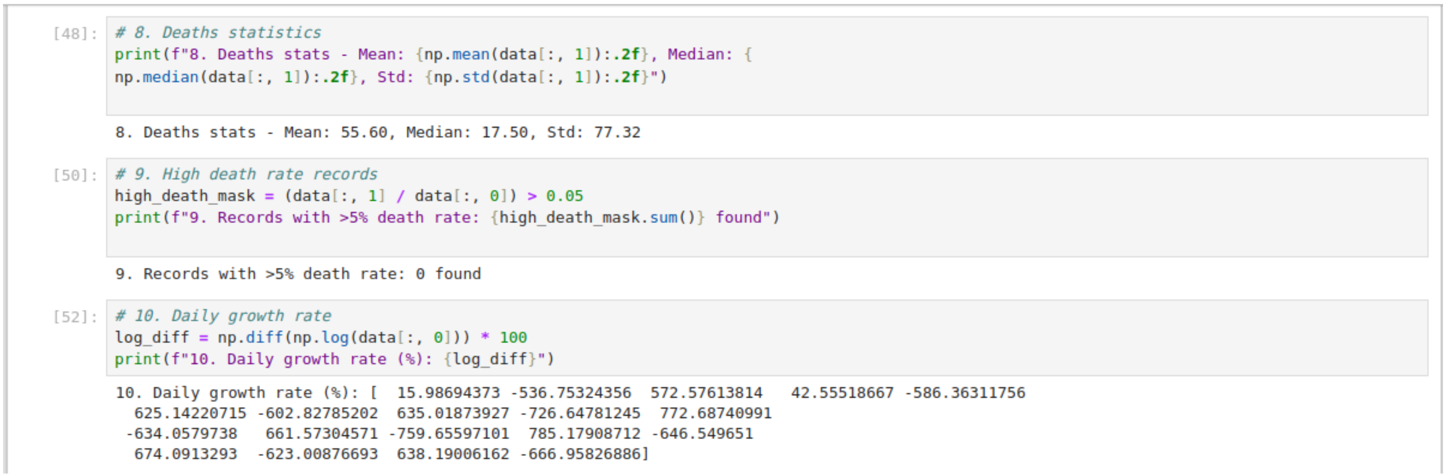
```
# 6. Normalized confirmed cases
confirmed_normalized = (data[:, 0] - np.min(data[:, 0])) / (np.max(data[:, 0]) - np.min(data[:, 0]))
print(f"6. Normalized confirmed cases: {confirmed_normalized}")
```

```
6. Normalized confirmed cases: [4.61968018e-02 5.42347068e-02 8.46095270e-05 7.76715458e-02
 1.18960995e-01 1.69219054e-04 1.75395550e-01 2.53828581e-04
 2.42067857e-01 0.00000000e+00 3.83704205e-01 5.07657162e-04
 5.05288095e-01 8.46095270e-05 6.52254844e-01 8.46095270e-04
 8.59125137e-01 1.52297149e-03 1.00000000e+00 1.09992385e-03]
```

[44]:

```
# 7. 7-day moving average for China
china_cases = np.array([item['New cases'] for item in covid_data if item['Country/Region'] == 'China'])
moving_avg = np.convolve(china_cases, np.ones(7)/7, mode='valid')
print(f"7. China's 7-day moving average: {moving_avg}")
```

```
7. China's 7-day moving average: [ 569.85714286  775.14285714 1009.71428571 1319.42857143 1487.57142857]
```



[48]:

```
# 8. Deaths statistics
print(f"8. Deaths stats - Mean: {np.mean(data[:, 1]):.2f}, Median: {
np.median(data[:, 1]):.2f}, Std: {np.std(data[:, 1]):.2f}")
```

```
8. Deaths stats - Mean: 55.60, Median: 17.50, Std: 77.32
```

[50]:

```
# 9. High death rate records
high_death_mask = (data[:, 1] / data[:, 0]) > 0.05
print(f"9. Records with >5% death rate: {high_death_mask.sum()} found")
```

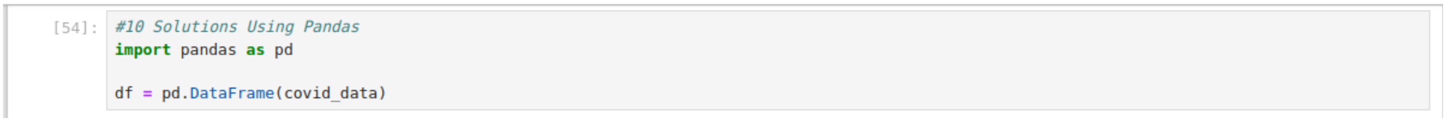
```
9. Records with >5% death rate: 0 found
```

[52]:

```
# 10. Daily growth rate
log_diff = np.diff(np.log(data[:, 0])) * 100
print(f"10. Daily growth rate (%): {log_diff}")
```

```
10. Daily growth rate (%): [  15.98694373 -536.75324356  572.57613814   42.55518667 -586.36311756
  625.14220715 -602.82785202  635.01873927 -726.64781245  772.68740991
 -634.0579738   661.57304571 -759.65597101  785.17908712 -646.549651
  674.0913293  -623.00876693  638.19006162 -666.95826886]
```

10 Problem statements and solution using Pandas



[54]:

```
#10 Solutions Using Pandas
import pandas as pd

df = pd.DataFrame(covid_data)
```

[56]:

```
# 1. Basic statistics
print("1. Dataset statistics:")
print(df.describe())
```

```
1. Dataset statistics:
          Confirmed      Deaths   Recovered        Active    New cases
count     20.000000   20.000000   20.000000     20.000000    20.000000
mean    2436.400000   55.600000   48.500000   2332.300000   565.000000
std     3670.206869   79.325047   68.685017   3523.314798   782.715381
min        2.000000    0.000000    0.000000      2.000000    -2.000000
25%        7.250000    0.000000    0.000000      7.250000     1.000000
50%      595.500000   17.500000   29.000000    549.000000    56.000000
75%     3281.500000   80.500000   59.500000   3141.500000   950.250000
max    11821.000000  259.000000  243.000000  11319.000000  2445.000000
```

[58]:

```
# 2. Top 3 countries by active cases
top3_active = df.groupby('Country/Region')['Active'].sum().nlargest(3)
print("\n2. Top 3 countries by active cases:")
print(top3_active)
```

```

2. Top 3 countries by active cases:
Country/Region
China    46575
US          40
Italy       31
Name: Active, dtype: int64
```

[60]:

```
# 3. Global recovery rate
recovery_rate = (df['Recovered'].sum() / df['Confirmed'].sum()) * 100
print(f"\n3. Global recovery rate: {recovery_rate:.2f}%")
```

```

3. Global recovery rate: 1.99%
```

[77]:

```
 # 4. Monthly deaths
df['Date'] = pd.to_datetime(df['Date'])
monthly_deaths = df.resample('M', on='Date')['Deaths'].sum()
print("\n4. Monthly deaths:")
print(monthly_deaths)
```

```

4. Monthly deaths:
Date
2020-01-31    231
2020-02-29    881
Freq: ME, Name: Deaths, dtype: int64
```

[64]:

```
# 5. Countries where cases doubled in a week
df['Weekly Growth'] = df.groupby('Country/Region')['New cases'].pct_change(periods=7)
doubled = df[df['Weekly Growth'] > 1]['Country/Region'].unique()
print("\n5. Countries with doubled cases in a week:")
print(doubled)
```

```

5. Countries with doubled cases in a week:
['China']
```

[66]:

```
# 6. Correlation matrix
corr_matrix = df[['Confirmed', 'Deaths', 'Recovered']].corr()
print("\n6. Correlation matrix:")
print(corr_matrix)
```

```

6. Correlation matrix:
           Confirmed    Deaths  Recovered
Confirmed   1.000000  0.998302   0.986056
Deaths      0.998302  1.000000   0.986917
Recovered   0.986056  0.986917   1.000000
```

[68]:

```
# 7. Italy's sorted data
italy_data = df[df['Country/Region'] == 'Italy'].sort_values('Date')
print("\n7. Italy's COVID-19 data:")
print(italy_data)
```

```

7. Italy's COVID-19 data:
         Date Country/Region  Confirmed  Deaths  Recovered  Active  New cases  \
2  2020-01-24          Italy          3       0          0       3          0
5  2020-01-27          Italy          4       0          0       4          1
9  2020-01-31          Italy          2       0          0       2         -2
13 2020-02-04          Italy          3       0          0       3          1
17 2020-02-08          Italy         20       1          0      19         17

    Weekly Growth
2             NaN
5             NaN
9             NaN
13            NaN
17            NaN
```

[70]:

```
# 8. Cumulative cases by country
df['Cumulative Confirmed'] = df.groupby('Country/Region')['Confirmed'].cumsum()
print("\n8. Cumulative cases sample:")
print(df[['Country/Region', 'Date', 'Cumulative Confirmed']].head())
```

```
8. Cumulative cases sample:
  Country/Region       Date  Cumulative Confirmed
0          China 2020-01-22                   548
1          China 2020-01-23                  1191
2          Italy 2020-01-24                     3
3          China 2020-01-25                  2111
4          China 2020-01-26                  3519
```

[72]:

```
# 9. Date with highest deaths
date_max_deaths = df.loc[df['Deaths'].idxmax(), 'Date']
print(f"\n9. Date with highest deaths: {date_max_deaths}")
```

```
9. Date with highest deaths: 2020-02-09 00:00:00
```

[74]:

```
# 10. Pivot table
pivot_table = df.pivot(index='Date', columns='Country/Region', values='Confirmed')
print("\n10. Pivot table (sample):")
print(pivot_table.head())
```

```
10. Pivot table (sample):
Country/Region   China  Italy  US
Date
2020-01-22       548.0    NaN NaN
2020-01-23       643.0    NaN NaN
2020-01-24         NaN    3.0 NaN
2020-01-25       920.0    NaN NaN
2020-01-26      1408.0    NaN NaN
```