

CSSE COVID-19 Dataset

เป็นข้อมูลสรุปรายงานการติดเชื้อ COVID-19 จากทั่วโลกตั้งแต่วันที่ 22 มกราคม 2020 ถึง 12 มีนาคม 2020

ประกอบด้วยไฟล์

- time_series_covid19_confirmed_global.csv
- time_series_covid19_deaths_global.csv
- time_series_covid19_recovered_global.csv

Field description

- Province/State:** China - province name; US/Canada/Australia/ - city name, state/province name; Others - name of the event (e.g., "Diamond Princess" cruise ship); other countries - blank.
- Country/Region:** country/region name conforming to WHO (will be updated).
- Confirmed:** the number of confirmed cases(accumulated).
- Deaths:** the number of deaths(accumulated).
- Recovered:** the number of recovered cases(accumulated).

จากไฟล์จึงทำการเขียนโปรแกรมเพื่อตอบคำถามต่อไปนี้

- จงแสดงอัตราการรักษาหายของเฉพาะผู้ป่วยในสหภาพยุโรปเป็นเปอร์เซ็นต์
- จงหา Country/Region ที่มียอดจำนวนผู้ติดเชื้อล่าสุดมากที่สุด 10 อันดับพร้อมพล็อตกราฟแสดงจำนวนผู้ติดเชื้อของแต่ละ Country/Region โดยเรียงลำดับจากมากไปน้อย (ไม่ต้องแยกเป็น Province/State)
- จงพล็อตกราฟเส้นเพื่อแสดง trend ยอดผู้ติดเชื้อในประเทศไทยในแต่ละสัปดาห์(กำหนดให้นับข้อมูลทุกวันอาทิตย์)
- จงหายอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซสพร้อมพล็อตกราฟ แล้วระบุวันที่มีการติดเชื้อมากที่สุด
- จงหา Country/Region ที่ล่าสุดมีเปอร์เซ็นต์อัตราการเสียชีวิตมากที่สุด 20 อันดับ พร้อมระบุว่า จาก Country/Region ในกลุ่มดังกล่าว ส่วนใหญ่อยู่ในทวีปอะไร
- จงหายอดผู้ป่วยที่กำลังรักษาตัวล่าสุด (สมมติว่ายังไม่เสียชีวิตหรือหายดี) ของแต่ละ Province/State ในประเทศจีน พร้อมพล็อตกราฟแสดงจำนวนในแต่ละ Province/State

1. จงแสดงอัตราการรักษาหายของเฉพาะผู้ป่วยในสหภาพยุโรปเป็นเปอร์เซ็นต์

```
1 #Solution
2 from google.colab import files
3 files.upload()

[Choose Files] time_series..._global.csv
• time_series_covid19_confirmed_global.csv(text/csv) - 67420 bytes, last modified: 4/13/2020 -
100% done
Saving time_series_covid19_confirmed_global.csv to time_series_covid19_confirmed_global.csv
{'time_series_covid19_confirmed_global.csv': b'Province/State,Country/Region,Lat,Long,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/20,1/29/20,1/30/20,1/31/20,2/1/20,2/2/20,2/3/20,2/4/20,2/5/20,2/6/20,2/7/20,2/8/20,2/9/20,2/10/20,2/11/20,2/12/20,2/13/20,2/14/20,2/15/20,2/16/20,2/17/20,2/18/20,2/19/20,2/20/20,2/21/20,2/22/20,2/23/20,2/24/20,2/25/20,2/26/20,2/27/20,2/28/20,2/29/20,2/30/20,3/1/20,3/2/20,3/3/20,3/4/20,3/5/20,3/6/20,3/7/20,3/8/20,3/9/20,3/10/20,3/11/20,3/12/20,3/13/20,3/14/20,3/15/20,3/16/20,3/17/20,3/18/20,3/19/20,3/20/20,3/21/20,3/22/20,3/23/20,3/24/20,3/25/20,3/26/20,3/27/20,3/28/20,3/29/20,3/30/20,3/31/20,4/1/20,4/2/20,4/3/20,4/4/20,4/5/20,4/6/20,4/7/20,4/8/20,4/9/20,4/10/20,4/11/20,4/12/20,4/13/20,4/14/20,4/15/20,4/16/20,4/17/20,4/18/20,4/19/20,4/20/20,4/21/20,4/22/20,4/23/20,4/24/20,4/25/20,4/26/20,4/27/20,4/28/20,4/29/20,4/30/20,5/1/20,5/2/20,5/3/20,5/4/20,5/5/20,5/6/20,5/7/20,5/8/20,5/9/20,5/10/20,5/11/20,5/12/20,5/13/20,5/14/20,5/15/20,5/16/20,5/17/20,5/18/20,5/19/20,5/20/20,5/21/20,5/22/20,5/23/20,5/24/20,5/25/20,5/26/20,5/27/20,5/28/20,5/29/20,5/30/20,5/31/20,6/1/20,6/2/20,6/3/20,6/4/20,6/5/20,6/6/20,6/7/20,6/8/20,6/9/20,6/10/20,6/11/20,6/12/20,6/13/20,6/14/20,6/15/20,6/16/20,6/17/20,6/18/20,6/19/20,6/20/20,6/21/20,6/22/20,6/23/20,6/24/20,6/25/20,6/26/20,6/27/20,6/28/20,6/29/20,6/30/20,7/1/20,7/2/20,7/3/20,7/4/20,7/5/20,7/6/20,7/7/20,7/8/20,7/9/20,7/10/20,7/11/20,7/12/20,7/13/20,7/14/20,7/15/20,7/16/20,7/17/20,7/18/20,7/19/20,7/20/20,7/21/20,7/22/20,7/23/20,7/24/20,7/25/20,7/26/20,7/27/20,7/28/20,7/29/20,7/30/20,7/31/20,8/1/20,8/2/20,8/3/20,8/4/20,8/5/20,8/6/20,8/7/20,8/8/20,8/9/20,8/10/20,8/11/20,8/12/20,8/13/20,8/14/20,8/15/20,8/16/20,8/17/20,8/18/20,8/19/20,8/20/20,8/21/20,8/22/20,8/23/20,8/24/20,8/25/20,8/26/20,8/27/20,8/28/20,8/29/20,8/30/20,8/31/20,9/1/20,9/2/20,9/3/20,9/4/20,9/5/20,9/6/20,9/7/20,9/8/20,9/9/20,9/10/20,9/11/20,9/12/20,9/13/20,9/14/20,9/15/20,9/16/20,9/17/20,9/18/20,9/19/20,9/20/20,9/21/20,9/22/20,9/23/20,9/24/20,9/25/20,9/26/20,9/27/20,9/28/20,9/29/20,9/30/20,10/1/20,10/2/20,10/3/20,10/4/20,10/5/20,10/6/20,10/7/20,10/8/20,10/9/20,10/10/20,10/11/20,10/12/20,10/13/20,10/14/20,10/15/20,10/16/20,10/17/20,10/18/20,10/19/20,10/20/20,10/21/20,10/22/20,10/23/20,10/24/20,10/25/20,10/26/20,10/27/20,10/28/20,10/29/20,10/30/20,10/31/20,11/1/20,11/2/20,11/3/20,11/4/20,11/5/20,11/6/20,11/7/20,11/8/20,11/9/20,11/10/20,11/11/20,11/12/20,11/13/20,11/14/20,11/15/20,11/16/20,11/17/20,11/18/20,11/19/20,11/20/20,11/21/20,11/22/20,11/23/20,11/24/20,11/25/20,11/26/20,11/27/20,11/28/20,11/29/20,11/30/20,12/1/20,12/2/20,12/3/20,12/4/20,12/5/20,12/6/20,12/7/20,12/8/20,12/9/20,12/10/20,12/11/20,12/12/20,12/13/20,12/14/20,12/15/20,12/16/20,12/17/20,12/18/20,12/19/20,12/20/20,12/21/20,12/22/20,12/23/20,12/24/20,12/25/20,12/26/20,12/27/20,12/28/20,12/29/20,12/30/20,12/31/20'}

1 files.upload()

[Choose Files] time_series..._global.csv
• time_series_covid19_deaths_global.csv(text/csv) - 54349 bytes, last modified: 4/13/2020 -
100% done
Saving time_series_covid19_deaths_global.csv to time_series_covid19_deaths_global.csv
{'time_series_covid19_deaths_global.csv': b'Province/State,Country/Region,Lat,Long,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/20,1/29/20,1/30/20,1/31/20,2/1/20,2/2/20,2/3/20,2/4/20,2/5/20,2/6/20,2/7/20,2/8/20,2/9/20,2/10/20,2/11/20,2/12/20,2/13/20,2/14/20,2/15/20,2/16/20,2/17/20,2/18/20,2/19/20,2/20/20,2/21/20,2/22/20,2/23/20,2/24/20,2/25/20,2/26/20,2/27/20,2/28/20,2/29/20,2/30/20,3/1/20,3/2/20,3/3/20,3/4/20,3/5/20,3/6/20,3/7/20,3/8/20,3/9/20,3/10/20,3/11/20,3/12/20,3/13/20,3/14/20,3/15/20,3/16/20,3/17/20,3/18/20,3/19/20,3/20/20,3/21/20,3/22/20,3/23/20,3/24/20,3/25/20,3/26/20,3/27/20,3/28/20,3/29/20,3/30/20,3/31/20,4/1/20,4/2/20,4/3/20,4/4/20,4/5/20,4/6/20,4/7/20,4/8/20,4/9/20,4/10/20,4/11/20,4/12/20,4/13/20,4/14/20,4/15/20,4/16/20,4/17/20,4/18/20,4/19/20,4/20/20,4/21/20,4/22/20,4/23/20,4/24/20,4/25/20,4/26/20,4/27/20,4/28/20,4/29/20,4/30/20,5/1/20,5/2/20,5/3/20,5/4/20,5/5/20,5/6/20,5/7/20,5/8/20,5/9/20,5/10/20,5/11/20,5/12/20,5/13/20,5/14/20,5/15/20,5/16/20,5/17/20,5/18/20,5/19/20,5/20/20,5/21/20,5/22/20,5/23/20,5/24/20,5/25/20,5/26/20,5/27/20,5/28/20,5/29/20,5/30/20,5/31/20,6/1/20,6/2/20,6/3/20,6/4/20,6/5/20,6/6/20,6/7/20,6/8/20,6/9/20,6/10/20,6/11/20,6/12/20,6/13/20,6/14/20,6/15/20,6/16/20,6/17/20,6/18/20,6/19/20,6/20/20,6/21/20,6/22/20,6/23/20,6/24/20,6/25/20,6/26/20,6/27/20,6/28/20,6/29/20,6/30/20,7/1/20,7/2/20,7/3/20,7/4/20,7/5/20,7/6/20,7/7/20,7/8/20,7/9/20,7/10/20,7/11/20,7/12/20,7/13/20,7/14/20,7/15/20,7/16/20,7/17/20,7/18/20,7/19/20,7/20/20,7/21/20,7/22/20,7/23/20,7/24/20,7/25/20,7/26/20,7/27/20,7/28/20,7/29/20,7/30/20,7/31/20,8/1/20,8/2/20,8/3/20,8/4/20,8/5/20,8/6/20,8/7/20,8/8/20,8/9/20,8/10/20,8/11/20,8/12/20,8/13/20,8/14/20,8/15/20,8/16/20,8/17/20,8/18/20,8/19/20,8/20/20,8/21/20,8/22/20,8/23/20,8/24/20,8/25/20,8/26/20,8/27/20,8/28/20,8/29/20,8/30/20,8/31/20,9/1/20,9/2/20,9/3/20,9/4/20,9/5/20,9/6/20,9/7/20,9/8/20,9/9/20,9/10/20,9/11/20,9/12/20,9/13/20,9/14/20,9/15/20,9/16/20,9/17/20,9/18/20,9/19/20,9/20/20,9/21/20,9/22/20,9/23/20,9/24/20,9/25/20,9/26/20,9/27/20,9/28/20,9/29/20,9/30/20,10/1/20,10/2/20,10/3/20,10/4/20,10/5/20,10/6/20,10/7/20,10/8/20,10/9/20,10/10/20,10/11/20,10/12/20,10/13/20,10/14/20,10/15/20,10/16/20,10/17/20,10/18/20,10/19/20,10/20/20,10/21/20,10/22/20,10/23/20,10/24/20,10/25/20,10/26/20,10/27/20,10/28/20,10/29/20,10/30/20,10/31/20,11/1/20,11/2/20,11/3/20,11/4/20,11/5/20,11/6/20,11/7/20,11/8/20,11/9/20,11/10/20,11/11/20,11/12/20,11/13/20,11/14/20,11/15/20,11/16/20,11/17/20,11/18/20,11/19/20,11/20/20,11/21/20,11/22/20,11/23/20,11/24/20,11/25/20,11/26/20,11/27/20,11/28/20,11/29/20,11/30/20,12/1/20,12/2/20,12/3/20,12/4/20,12/5/20,12/6/20,12/7/20,12/8/20,12/9/20,12/10/20,12/11/20,12/12/20,12/13/20,12/14/20,12/15/20,12/16/20,12/17/20,12/18/20,12/19/20,12/20/20,12/21/20,12/22/20,12/23/20,12/24/20,12/25/20,12/26/20,12/27/20,12/28/20,12/29/20,12/30/20,12/31/20'}

1 files.upload()

[Choose Files] time_series..._global.csv
• time_series_covid19_recovered_global.csv(text/csv) - 56755 bytes, last modified: 4/13/2020 -
100% done
Saving time_series_covid19_recovered_global.csv to time_series_covid19_recovered_global.csv
{'time_series_covid19_recovered_global.csv': b'Province/State,Country/Region,Lat,Long,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/20,1/29/20,1/30/20,1/31/20,2/1/20,2/2/20,2/3/20,2/4/20,2/5/20,2/6/20,2/7/20,2/8/20,2/9/20,2/10/20,2/11/20,2/12/20,2/13/20,2/14/20,2/15/20,2/16/20,2/17/20,2/18/20,2/19/20,2/20/20,2/21/20,2/22/20,2/23/20,2/24/20,2/25/20,2/26/20,2/27/20,2/28/20,2/29/20,2/30/20,3/1/20,3/2/20,3/3/20,3/4/20,3/5/20,3/6/20,3/7/20,3/8/20,3/9/20,3/10/20,3/11/20,3/12/20,3/13/20,3/14/20,3/15/20,3/16/20,3/17/20,3/18/20,3/19/20,3/20/20,3/21/20,3/22/20,3/23/20,3/24/20,3/25/20,3/26/20,3/27/20,3/28/20,3/29/20,3/30/20,3/31/20,4/1/20,4/2/20,4/3/20,4/4/20,4/5/20,4/6/20,4/7/20,4/8/20,4/9/20,4/10/20,4/11/20,4/12/20,4/13/20,4/14/20,4/15/20,4/16/20,4/17/20,4/18/20,4/19/20,4/20/20,4/21/20,4/22/20,4/23/20,4/24/20,4/25/20,4/26/20,4/27/20,4/28/20,4/29/20,4/30/20,5/1/20,5/2/20,5/3/20,5/4/20,5/5/20,5/6/20,5/7/20,5/8/20,5/9/20,5/10/20,5/11/20,5/12/20,5/13/20,5/14/20,5/15/20,5/16/20,5/17/20,5/18/20,5/19/20,5/20/20,5/21/20,5/22/20,5/23/20,5/24/20,5/25/20,5/26/20,5/27/20,5/28/20,5/29/20,5/30/20,5/31/20,6/1/20,6/2/20,6/3/20,6/4/20,6/5/20,6/6/20,6/7/20,6/8/20,6/9/20,6/10/20,6/11/20,6/12/20,6/13/20,6/14/20,6/15/20,6/16/20,6/17/20,6/18/20,6/19/20,6/20/20,6/21/20,6/22/20,6/23/20,6/24/20,6/25/20,6/26/20,6/27/20,6/28/20,6/29/20,6/30/20,7/1/20,7/2/20,7/3/20,7/4/20,7/5/20,7/6/20,7/7/20,7/8/20,7/9/20,7/10/20,7/11/20,7/12/20,7/13/20,7/14/20,7/15/20,7/16/20,7/17/20,7/18/20,7/19/20,7/20/20,7/21/20,7/22/20,7/23/20,7/24/20,7/25/20,7/26/20,7/27/20,7/28/20,7/29/20,7/30/20,7/31/20,8/1/20,8/2/20,8/3/20,8/4/20,8/5/20,8/6/20,8/7/20,8/8/20,8/9/20,8/10/20,8/11/20,8/12/20,8/13/20,8/14/20,8/15/20,8/16/20,8/17/20,8/18/20,8/19/20,8/20/20,8/21/20,8/22/20,8/23/20,8/24/20,8/25/20,8/26/20,8/27/20,8/28/20,8/29/20,8/30/20,8/31/20,9/1/20,9/2/20,9/3/20,9/4/20,9/5/20,9/6/20,9/7/20,9/8/20,9/9/20,9/10/20,9/11/20,9/12/20,9/13/20,9/14/20,9/15/20,9/16/20,9/17/20,9/18/20,9/19/20,9/20/20,9/21/20,9/22/20,9/23/20,9/24/20,9/25/20,9/26/20,9/27/20,9/28/20,9/29/20,9/30/20,10/1/20,10/2/20,10/3/20,10/4/20,10/5/20,10/6/20,10/7/20,10/8/20,10/9/20,10/10/20,10/11/20,10/12/20,10/13/20,10/14/20,10/15/20,10/16/20,10/17/20,10/18/20,10/19/20,10/20/20,10/21/20,10/22/20,10/23/20,10/24/20,10/25/20,10/26/20,10/27/20,10/28/20,10/29/20,10/30/20,10/31/20,11/1/20,11/2/20,11/3/20,11/4/20,11/5/20,11/6/20,11/7/20,11/8/20,11/9/20,11/10/20,11/11/20,11/12/20,11/13/20,11/14/20,11/15/20,11/16/20,11/17/20,11/18/20,11/19/20,11/20/20,11/21/20,11/22/20,11/23/20,11/24/20,11/25/20,11/26/20,11/27/20,11/28/20,11/29/20,11/30/20,12/1/20,12/2/20,12/3/20,12/4/20,12/5/20,12/6/20,12/7/20,12/8/20,12/9/20,12/10/20,12/11/20,12/12/20,12/13/20,12/14/20,12/15/20,12/16/20,12/17/20,12/18/20,12/19/20,12/20/20,12/21/20,12/22/20,12/23/20,12/24/20,12/25/20,12/26/20,12/27/20,12/28/20,12/29/20,12/30/20,12/31/20'}

1 pip install pycountry_convert

[Choose Files] time_series..._global.csv
```

Requirement already satisfied: pycountry_convert in /usr/local/lib/python3.6/dist-packages (0.7.2)
Requirement already satisfied: pprintpp>=0.3.0 in /usr/local/lib/python3.6/dist-packages (from pycountry_convert) (0.4.0)
Requirement already satisfied: pycountry>=16.11.27.1 in /usr/local/lib/python3.6/dist-packages (from pycountry_convert) (19.8.0)
Requirement already satisfied: pytest>=3.4.0 in /usr/local/lib/python3.6/dist-packages (from pycountry_convert) (3.6.4)
Requirement already satisfied: pytest-cov>=2.5.1 in /usr/local/lib/python3.6/dist-packages (from pycountry_convert) (2.10.0)
Requirement already satisfied: reparse-logs>=0.7 in /usr/local/lib/python3.6/dist-packages (from pycountry_convert) (0.7)

```
1 import pandas as pd
2 import pycountry_convert as pc
3 cf = pd.read_csv('time_series_covid19_confirmed_global.csv', delimiter = ',')
4 dt = pd.read_csv('time_series_covid19_deaths_global.csv', delimiter = ',')
5 rc = pd.read_csv('time_series_covid19_recovered_global.csv', delimiter = ',')

        requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from pytest>=3.4.0->pycountry_convert) (45.2.0)

1 cf['Country/Region'] = cf['Country/Region'].replace(
2     ['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Cote d'Ivoire', 'Burma', 'Korea, South', 'Taiwan*', 'US*'],
3     ['Congo', 'Congo', 'Ivory Coast', 'Myanmar', 'South Korea', 'Taiwan', 'United States'])
4
5
6 rc['Country/Region'] = rc['Country/Region'].replace(
7     ['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Cote d'Ivoire', 'Burma', 'Korea, South', 'Taiwan*', 'US*'],
8     ['Congo', 'Congo', 'Ivory Coast', 'Myanmar', 'South Korea', 'Taiwan', 'United States'])
9
10
11 dt['Country/Region'] = dt['Country/Region'].replace(
12     ['Congo (Brazzaville)', 'Congo (Kinshasa)', 'Cote d'Ivoire', 'Burma', 'Korea, South', 'Taiwan*', 'US*'],
13     ['Congo', 'Congo', 'Ivory Coast', 'Myanmar', 'South Korea', 'Taiwan', 'United States'])
14
```

```
1 country_list = cf['Country/Region'].unique()
2 len(country_list)
```

➞ 184

```
1 convert = {}
2
3 for country in country_list:
4     try :
5         convert[country] = pc.convert_continent_code_to_continent_name(pc.country_alpha2_to_continent_code(pc.country_name_to_continent_code(country)))
6     except :
7         convert[country] = "Other"
8
9 len(convert)
```

➞ 184

```
1 cf['Continents'] = cf['Country/Region'].map(convert)
2 dt['Continents'] = dt['Country/Region'].map(convert)
3 rc['Continents'] = rc['Country/Region'].map(convert)
```

```
1 rc_result = rc.drop(axis=1, columns=['Province/State', 'Lat', 'Long'])
2 rc_max = rc_result.groupby(['Country/Region']).agg('sum')
3 rc_max = rc_max['4/12/20'].to_frame('Max_Recover')
4 cf_result = cf.drop(axis=1, columns=['Province/State', 'Lat', 'Long'])
5 cf_max = cf_result.groupby(['Country/Region']).agg('sum')
6 cf_max = cf_max['4/12/20'].to_frame('Max_Confirmed')
7 dt_result = dt.drop(axis=1, columns=['Province/State', 'Lat', 'Long'])
8 dt_max = dt_result.groupby(['Country/Region']).agg('sum')
9 dt_max = dt_max['4/12/20'].to_frame('Max_Death')
```

```
1 eu_list = ['Austria', 'Belgium', 'Bulgaria', 'Croatia', 'Cyprus', 'Czechia', 'Denmark', 'Estonia', 'Finland', 'France', 'Germany', 'Greece', 'Hungary', 'Ireland', 'Italy', 'Latvia', 'Lithuania', 'Luxembourg', 'Malta', 'Netherlands', 'Poland', 'Portugal', 'Romania', 'Slovakia', 'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Turkey', 'Ukraine', 'United Kingdom', 'United States']
2 x = rc_max.reset_index()
3 rc_eu = x[x['Country/Region'].isin(eu_list)]
4 y = cf_max.reset_index()
5 cf_eu = y[y['Country/Region'].isin(eu_list)]
6
7 eu = pd.merge(rc_eu, cf_eu, how='inner', left_on='Country/Region', right_on='Country/Region')
8 eu['Union'] = 'European Union'
9 eu = eu.groupby(['Union']).agg('sum')
10 eu['Percent_recovered'] = eu['Max_Recover'] / eu['Max_Confirmed'] * 100
11
12 print(eu['Percent_recovered'])
13 print ("\nอัตราการรักษาหายของผู้ป่วยในสหภาพยุโรปเท่ากับ " + " " + str(eu.loc['European Union', 'Percent_recovered']) + "%")
```

➞ Union
European Union 28.003272
Name: Percent_recovered, dtype: float64

อัตราการรักษาหายของผู้ป่วยในสหภาพยุโรปเท่ากับ 28.003272111416216%

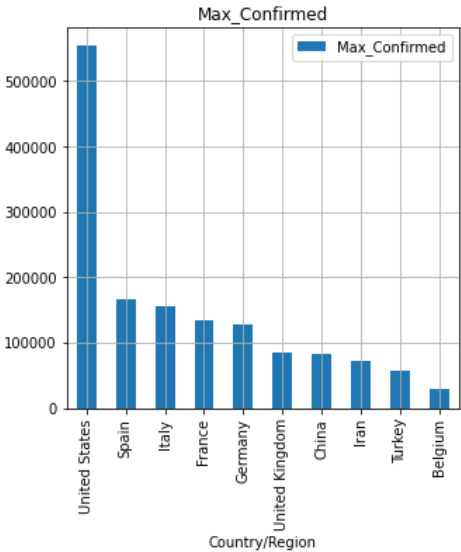
2. จงหา Country/Region ที่มียอดจำนวนผู้ติดเชื้อล่าสุดมากที่สุด 10 อันดับพร้อมพล็อตกราฟแสดงจำนวนผู้ติดเชื้อของแต่ละ Country/Region โดยเรียงลำดับจากมากไปน้อย (ไม่ต้องแยกเป็น Province/State)

```
1 #Solution
2 top10_cf = cf_result.groupby(['Country/Region']).agg('sum')
3 top10_cf = top10_cf.nlargest(10, '4/12/20')
4 top10_cf = top10_cf['4/12/20'].to_frame('Max_Confirmed')
5
6 print('Country/Region ที่มียอดจำนวนผู้ติดเชื้อล่าสุดมากที่สุด 10 อันดับ\n')
7 print(top10_cf)
8 print("\n")
9
10 top10_cf.plot(kind='bar', subplots=True, figsize=(5, 5), legend=True, grid=True)
```

➡ Country/Region ที่มียอดจำนวนผู้ติดเชื้อล่าสุดมากที่สุด 10 อันดับ

	Max_Confirmed
Country/Region	
United States	555313
Spain	166831
Italy	156363
France	133670
Germany	127854
United Kingdom	85206
China	83134
Iran	71686
Turkey	56956
Belgium	29647

array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fb79dac2780>],
dtype=object)



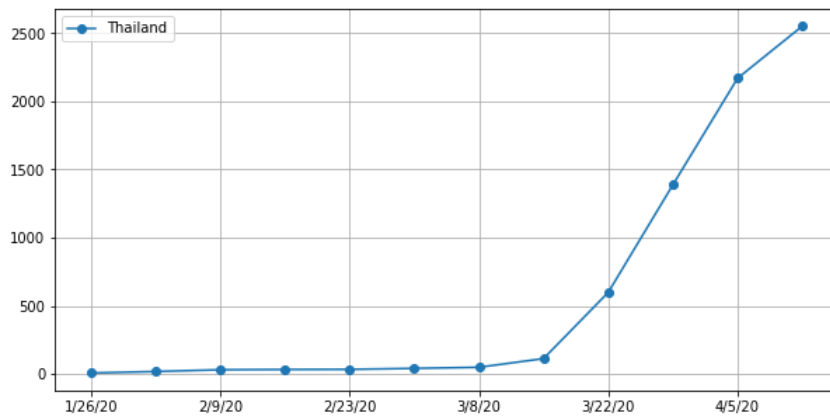
3. จงพล็อตกราฟเส้นเพื่อแสดง trend ยอดผู้ติดเชื้อในประเทศไทยในแต่ละสัปดาห์(กำหนดให้นับข้อมูลทุกวันอาทิตย์)

```
1 #Solution
2 thai = cf_result.groupby('Country/Region').agg('sum')
3 thai = thai.T['Thailand'].to_frame('Thailand')
4 thai = thai.reset_index()
5 datetime = pd.to_datetime(thai['index'])
6 dayname = datetime.dt.day_name()
7 thai['Day_Name'] = dayname
8
9 #print(thai)
10 #print("\n")
11 thai_sunday = thai[thai['Day_Name'] == 'Sunday']
12
13 #print(thai_sunday)
14 #print("\n")
15 print('กราฟเส้นแสดง trend ยอดผู้ติดเชื้อในประเทศไทยแต่ละสัปดาห์ (ทุกวันอาทิตย์)\n')
16
17 line_chart = thai_sunday.drop(axis=1, columns=['Day_Name'])
18 line_chart.plot.line(x='index', y='Thailand', figsize=(10, 5), legend=True, grid=True, linestyle='-', marker='o')
```

➡

กราฟแสดง trend ยอดผู้ติดเชื้อในประเทศไทยแต่ละสัปดาห์ (ทุกวันอาทิตย์)

<matplotlib.axes._subplots.AxesSubplot at 0x7fb79db2c748>



▼ 4. จงหายอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซสพร้อมพล็อตกราฟ แล้วระบุวันที่มีการติดเชื้อมากที่สุด

```
1 #Solution
2 diamond = cf_result.groupby('Country/Region').agg('sum')
3 diamond = diamond.T['Diamond Princess'].to_frame('Diamond_Princess')
4
5 #print(diamond)
6 #print('\n')
7
8
9 plot_diamond = diamond.diff()
10 plot_diamond['Diamond_Princess'] = plot_diamond['Diamond_Princess'].replace([None], [0])
11
12 print('ยอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซส\n')
13 print(plot_diamond)
14 print('\n')
15 print('กราฟแสดงยอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซส\n')
16
17 plot_diamond.plot(kind='line', figsize=(10, 5), legend=True, grid=True, linestyle='-', marker='o')
```

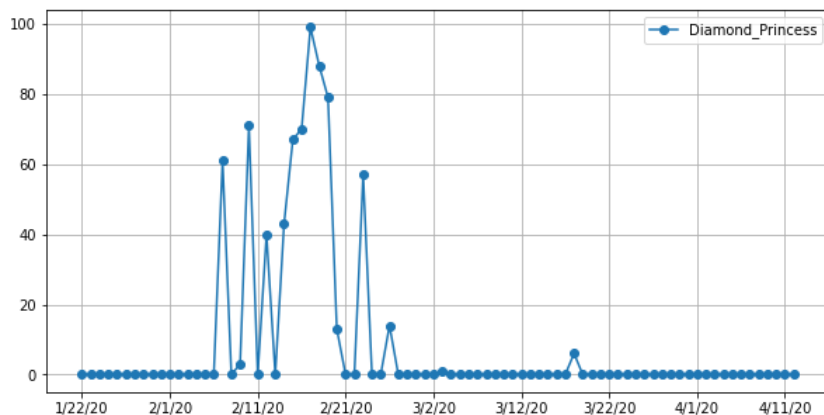
➤ ยอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซส

	Diamond_Princess
1/22/20	0.0
1/23/20	0.0
1/24/20	0.0
1/25/20	0.0
1/26/20	0.0
...	...
4/8/20	0.0
4/9/20	0.0
4/10/20	0.0
4/11/20	0.0
4/12/20	0.0

[82 rows x 1 columns]

กราฟแสดงยอดผู้ติดเชื้อของแต่ละวันบนเรือไดมอนด์พริ้นเซส

<matplotlib.axes._subplots.AxesSubplot at 0x7fb79d88ac88>



```
1 plot_diamond = plot_diamond.idxmax(axis=0, skipna=True)
2 print('วันที่มีการติดเชื้อมากที่สุด คือ วันที่ ' + plot_diamond)
```

➤ Diamond_Princess วันที่มีการติดเชื้อมากที่สุด คือ วันที่ 2/17/20
dtype: object

5. จงหา Country/Region ที่ล่าสุดมีเปอร์เซ็นต์อัตราการเสียชีวิตมากที่สุด 20 อันดับ พร้อมระบุว่า จาก Country/Region ในกลุ่มดังกล่าว ส่วนใหญ่อยู่ในทวีปอะไร

```
1 #Solution
2 max_dt = dt_result.groupby(['Country/Region']).agg('sum')
3 max_cf = cf_result.groupby(['Country/Region']).agg('sum')
4
5 mdt = max_dt['4/12/20'].to_frame('Max_Death')
6 mcf = max_cf['4/12/20'].to_frame('Max_Confirmed')
7
8 mdt['Max_Confirmed'] = mcf['Max_Confirmed']
9
10 #print(mdt)
11 #print('\n')
12 print('Country/Region ที่มีอัตราเสียชีวิตมากที่สุด 20 อันดับ\n')
13
14 mdt['Percent_Death'] = mdt['Max_Death'] / mdt['Max_Confirmed'] * 100
15
16 mdt = mdt.reset_index()
17 mdt['Continents'] = mdt['Country/Region'].map(convert)
18 mdt
19
20 top20_dt = mdt.nlargest(20, 'Percent_Death')
21 print(top20_dt[['Country/Region', 'Continents', 'Percent_Death']])
```

➞ Country/Region ที่มีอัตราเสียชีวิตมากที่สุด 20 อันดับ

	Country/Region	Continents	Percent_Death
99	MS Zaandam	Other	22.222222
183	Zimbabwe	Africa	21.428571
11	Bahamas	North America	17.391304
101	Malawi	Africa	15.384615
2	Algeria	Africa	15.308255
17	Belize	North America	14.285714
106	Mauritania	Africa	14.285714
69	Guyana	South America	13.333333
81	Italy	Europe	12.726156
28	Cabo Verde	Africa	12.500000
173	United Kingdom	Europe	12.474474
16	Belgium	Europe	12.142881
60	Gambia	Africa	11.111111
120	Nicaragua	North America	11.111111
58	France	Europe	10.781776
118	Netherlands	Europe	10.669619
4	Angola	Africa	10.526316
157	Sudan	Africa	10.526316
155	Spain	Europe	10.315229
94	Liberia	Africa	10.000000

```
1 count_continents = top20_dt.pivot_table(index=['Continents'], aggfunc='size')
2 count_continents = count_continents.idxmax(axis=0, skipna=True)
3 print('ทวีปที่มีอัตราการเสียชีวิตมากที่สุด คือ ' + count_continents)
```

➞ ทวีปที่มีอัตราการเสียชีวิตมากที่สุด คือ Africa

6. จงหายอดผู้ป่วยที่กำลังรักษาตัวล่าสุด (สมมติว่ายังไม่เสียชีวิตหรือหายดี) ของแต่ละ Province/State ใน ประเทศจีน พร้อมพล็อตกราฟแสดงจำนวนในแต่ละ Province/State

```
1 #Solution
2 chinarc = rc.groupby(['Country/Region', 'Province/State'])[['Country/Region', 'Province/State', '4/12/20']].agg('sum')
3 chinarc = chinarc.reset_index()
4 chinarc = chinarc[(chinarc['Country/Region'] == 'China')][["Province/State", 'Country/Region', '4/12/20']]
5
6
7 chinacf = cf.groupby(['Country/Region', 'Province/State'])[['Country/Region', 'Province/State', '4/12/20']].agg('sum')
8 chinacf = chinacf.reset_index()
9 chinacf = chinacf[(chinacf['Country/Region'] == 'China')][["Province/State", 'Country/Region', '4/12/20']]
10
11
12 chinadt = dt.groupby(['Country/Region', 'Province/State'])[['Country/Region', 'Province/State', '4/12/20']].agg('sum')
13 chinadt = chinadt.reset_index()
14 chinadt = chinadt[(chinadt['Country/Region'] == 'China')][["Province/State", 'Country/Region', '4/12/20']]
15
16
17 #chinarc.shape
18 #chinacf.shape
19 #chinadt.shape
20
```

```
21 chinarc = chinarc.drop(axis=1, columns=['Country/Region'])
22 chinadt = chinadt.drop(axis=1, columns=['Country/Region'])
23
24 china = pd.merge(chinacf, chinarc, how='inner', left_on='Province/State', right_on='Province/State')
25 china = pd.merge(china, chinadt, how='inner', left_on='Province/State', right_on='Province/State')
26 china.rename(columns={'4/12/20_x' : 'Confirmed', '4/12/20_y' : 'Recovered', '4/12/20' : 'Death'}, inplace=True)
27 china['In_Process'] = china['Confirmed'] - china['Recovered'] - china['Death']
28 china_inprocess = china[['Province/State', 'In_Process']]
29
30 print('ยอดผู้ป่วยที่กำลังรักษาตัวล่าสุด ของแต่ละ Province/State ในประเทศจีน\n')
31 print(china_inprocess)
32 print('\n')
33
34 china_inprocess.plot.bar(x='Province/State', y='In_Process', figsize=(15, 5), legend=True, grid=True)
```

➡ ยอดผู้ป่วยที่กำลังรักษาตัวล่าสุด ของแต่ละ Province/State ในประเทศจีน

	Province/State	In_Process
0	Anhui	1
1	Beijing	102
2	Chongqing	3
3	Fujian	30
4	Gansu	2
5	Guangdong	101
6	Guangxi	0
7	Guizhou	0
8	Hainan	0
9	Hebei	8
10	Heilongjiang	201
11	Henan	1
12	Hong Kong	640
13	Hubei	303
14	Hunan	1
15	Inner Mongolia	105
16	Jiangsu	15
17	Jiangxi	0
18	Jilin	2
19	Liaoning	7
20	Macao	32
21	Ningxia	0
22	Qinghai	0
23	Shaanxi	5
24	Shandong	20
25	Shanghai	154
26	Shanxi	37
27	Sichuan	10
28	Tianjin	19
29	Tibet	0
30	Xinjiang	0
31	Yunnan	8
32	Zhejiang	28

<matplotlib.axes._subplots.AxesSubplot at 0x7fb79e6d7a58>

