In [1]:
```
# import libraries
# for date and time opeations
from datetime import datetime, timedelta
# storing and analysing data
import pandas as pd
# numerical analysis
import numpy as np
```

In [2]:
```
# urls of the files
urls = ['https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_
        'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_
        'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_

confirmed_global=pd.read_csv(urls[0])
recovered_global=pd.read_csv(urls[2])
deaths_global=pd.read_csv(urls[1])
```

In [3]:
```
confirmed_global.head()
```

Out[3]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	

5 rows × 1122 columns

Merging Data frames

In [4]:
```
# extract dates
dates = confirmed_global.columns[4:]
dates
```

Out[4]:
```
Index(['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',
       ...
       '2/3/23', '2/4/23', '2/5/23', '2/6/23', '2/7/23', '2/8/23', '2/9/23',
       '2/10/23', '2/11/23', '2/12/23'],
      dtype='object', length=1118)
```

In [51]:
```
# melt dataframes into longer format
conf_df_long = confirmed_global.melt(id_vars=['Province/State', 'Country/Region', 'Lat',
                                              value_vars=dates, var_name='Date', value_name='Confirmed')

deaths_df_long = deaths_global.melt(id_vars=['Province/State', 'Country/Region', 'Lat',
                                              value_vars=dates, var_name='Date', value_name='Deaths')

recv_df_long = recovered_global.melt(id_vars=['Province/State', 'Country/Region', 'Lat',
                                              value_vars=dates, var_name='Date', value_name='Recovered')

recv_df_long = recv_df_long[recv_df_long['Country/Region']!='Canada']

print(conf_df_long.shape)
print(deaths_df_long.shape)
print(recv_df_long.shape)
```

```
(323102, 6)
(323102, 6)
(305214, 6)
```

In [ ]:

In [52]:
```
# merge dataframes

full_table = pd.merge(left=conf_df_long, right=deaths_df_long, how='left',
                      on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long'])
full_table = pd.merge(left=full_table, right=recv_df_long, how='left',
                      on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long'])

full_table.head()
```

Out[52]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0	0	0.0
1	NaN	Albania	41.15330	20.168300	1/22/20	0	0	0.0
2	NaN	Algeria	28.03390	1.659600	1/22/20	0	0	0.0
3	NaN	Andorra	42.50630	1.521800	1/22/20	0	0	0.0
4	NaN	Angola	-11.20270	17.873900	1/22/20	0	0	0.0

In [53]:
```
full_table.sample(10)
```

Out[53]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
321925	Channel Islands	United Kingdom	49.372300	-2.364400	2/8/23	0	0	0.0
75101	NaN	Tajikistan	38.861000	71.276100	10/7/20	10014	78	8876.0
180015	NaN	Tunisia	33.886917	9.537499	10/5/21	708788	24966	0.0
148907	Hubei	China	30.975600	112.270700	6/20/21	68160	4512	63647.0
21157	Beijing	China	40.182400	116.414200	4/4/20	585	8	438.0
137868	Victoria	Australia	-37.813600	144.963100	5/13/21	20539	820	19701.0
268112	NaN	Oman	21.512583	55.923255	8/6/22	396722	4628	0.0
254072	British Columbia	Canada	53.726700	-127.647600	6/19/22	373336	3682	NaN
290711	British Virgin Islands	United Kingdom	18.420700	-64.640000	10/23/22	7305	64	0.0
101738	New South Wales	Australia	-33.868800	151.209300	1/8/21	5001	54	0.0

Fixing ambigious data in recovered column

In [54]:
```
temp = full_table.groupby(by=['Country/Region']).sum()
condition =temp['Recovered'] == 0

temp[condition].shape
```

Out[54]:
```
(14, 5)
```

In [55]:
```
full_table['Recovered/Active'] = full_table['Confirmed'] - full_table['Deaths']
full_table.sample(10)
```

Out[55]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
262282	NaN	Kenya	-0.023600	37.906200	7/17/22	336740	5668	0.0
294716	NaN	Saint Vincent and the Grenadines	12.984300	-61.287200	11/6/22	9459	116	0.0
258772	NaN	Ethiopia	9.145000	40.489700	7/5/22	489656	7544	0.0
232559	NaN	New Zealand	-40.900600	174.886000	4/5/22	730285	434	0.0
195213	NaN	Grenada	12.116500	-61.679000	11/27/21	5888	200	0.0
221439	Guangxi	China	23.829800	108.788100	2/26/22	1079	2	0.0
27636	NaN	Malta	35.937500	14.375400	4/26/20	448	4	282.0
185027	Hainan	China	19.195900	109.745300	10/23/21	190	6	0.0
108185	NaN	Croatia	45.100000	15.200000	1/30/21	232090	4998	224017.0
48505	NaN	Sri Lanka	7.873054	80.771797	7/7/20	2081	11	1955.0

In [56]:
```
full_table.drop('Recovered',axis=1, inplace=True)
```

In [57]:
```
full_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 323102 entries, 0 to 323101
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   Province/State   101738 non-null  object 
 1   Country/Region   323102 non-null  object 
 2   Lat              320866 non-null  float64
 3   Long             320866 non-null  float64
 4   Date             323102 non-null  object 
 5   Confirmed        323102 non-null  int64  
 6   Deaths           323102 non-null  int64  
 7   Recovered/Active 323102 non-null  int64  
dtypes: float64(2), int64(3), object(3)
memory usage: 22.2+ MB
```

In [58]:
```
# Remove unused columns
full_table.drop(['Long', 'Lat'], axis=1, inplace=True)
```

In [59]:
```
full_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 323102 entries, 0 to 323101
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   Province/State   101738 non-null  object
 1   Country/Region   323102 non-null  object
 2   Date             323102 non-null  object
 3   Confirmed        323102 non-null  int64 
 4   Deaths           323102 non-null  int64 
 5   Recovered/Active 323102 non-null  int64 
dtypes: int64(3), object(3)
memory usage: 17.3+ MB
```

Data Processing

In [60]:
```
# Convert to proper date format
```

```
full_table['Date'] = pd.to_datetime(full_table['Date'])
```

In [61]:
```
#check country column
full_table['Country/Region'].unique()
```

Out[61]:
```
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antarctica', 'Antigua and Barbuda', 'Argentina', 'Armenia',
       'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
       'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin',
       'Bhutan', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana',
       'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
       'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
       'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia',
       'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica',
       "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
       'Diamond Princess', 'Djibouti', 'Dominica', 'Dominican Republic',
       'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
       'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France',
       'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
       'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana',
       'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
       'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
       'Korea, North', 'Korea, South', 'Kosovo', 'Kuwait', 'Kyrgyzstan',
       'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya',
       'Liechtenstein', 'Lithuania', 'Luxembourg', 'MS Zaandam',
       'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
       'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico',
       'Micronesia', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
       'Morocco', 'Mozambique', 'Namibia', 'Nauru', 'Nepal',
       'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria',
       'North Macedonia', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Panama',
       'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland',
       'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
       'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
       'Spain', 'Sri Lanka', 'Sudan', 'Summer Olympics 2020', 'Suriname',
       'Sweden', 'Switzerland', 'Syria', 'Taiwan*', 'Tajikistan',
       'Tanzania', 'Thailand', 'Timor-Leste', 'Togo', 'Tonga',
       'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Tuvalu', 'US',
       'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom',
       'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',
       'West Bank and Gaza', 'Winter Olympics 2022', 'Yemen', 'Zambia',
       'Zimbabwe'], dtype=object)
```

In [62]:
```
mask = full_table['Country/Region'].str.contains(",")
```

In [63]:
```
full_table[mask]['Country/Region'].unique()
```

Out[63]:
```
array(['Korea, North', 'Korea, South'], dtype=object)
```

In [66]:
```
full_table['Country/Region'] = full_table['Country/Region'].replace('Korea, South', 'Sou
full_table['Country/Region'] = full_table['Country/Region'].replace('Korea, North', 'Nor
con=full_table['Country/Region']=='Mainland China'
```

Out[66]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active

In [67]:
```
full_table['Country/Region'].value_counts()
```

Out[67]:
```
China              38012
```

```
Canada             17888
United Kingdom     16770
France             13416
Australia           8944
                    ...
Guinea              1118
Guinea-Bissau       1118
Guyana              1118
Haiti               1118
Zimbabwe            1118
Name: Country/Region, Length: 201, dtype: int64
```

In [68]:
```
full_table['Province/State'].unique()
```

Out[68]:
```
array([nan, 'Australian Capital Territory', 'New South Wales',
       'Northern Territory', 'Queensland', 'South Australia', 'Tasmania',
       'Victoria', 'Western Australia', 'Alberta', 'British Columbia',
       'Diamond Princess', 'Grand Princess', 'Manitoba', 'New Brunswick',
       'Newfoundland and Labrador', 'Northwest Territories',
       'Nova Scotia', 'Nunavut', 'Ontario', 'Prince Edward Island',
       'Quebec', 'Repatriated Travellers', 'Saskatchewan', 'Yukon',
       'Anhui', 'Beijing', 'Chongqing', 'Fujian', 'Gansu', 'Guangdong',
       'Guangxi', 'Guizhou', 'Hainan', 'Hebei', 'Heilongjiang', 'Henan',
       'Hong Kong', 'Hubei', 'Hunan', 'Inner Mongolia', 'Jiangsu',
       'Jiangxi', 'Jilin', 'Liaoning', 'Macau', 'Ningxia', 'Qinghai',
       'Shaanxi', 'Shandong', 'Shanghai', 'Shanxi', 'Sichuan', 'Tianjin',
       'Tibet', 'Unknown', 'Xinjiang', 'Yunnan', 'Zhejiang',
       'Faroe Islands', 'Greenland', 'French Guiana', 'French Polynesia',
       'Guadeloupe', 'Martinique', 'Mayotte', 'New Caledonia', 'Reunion',
       'Saint Barthelemy', 'Saint Pierre and Miquelon', 'St Martin',
       'Wallis and Futuna', 'Aruba', 'Bonaire, Sint Eustatius and Saba',
       'Curacao', 'Sint Maarten', 'Cook Islands', 'Niue', 'Anguilla',
       'Bermuda', 'British Virgin Islands', 'Cayman Islands',
       'Channel Islands', 'Falkland Islands (Malvinas)', 'Gibraltar',
       'Guernsey', 'Isle of Man', 'Jersey', 'Montserrat',
       'Pitcairn Islands', 'Saint Helena, Ascension and Tristan da Cunha',
       'Turks and Caicos Islands'], dtype=object)
```

In [69]:
```
full_table[full_table['Province/State']=='Greenland']
```

Out[69]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active
104	Greenland	Denmark	2020-01-22	0	0	0
393	Greenland	Denmark	2020-01-23	0	0	0
682	Greenland	Denmark	2020-01-24	0	0	0
971	Greenland	Denmark	2020-01-25	0	0	0
1260	Greenland	Denmark	2020-01-26	0	0	0
...	...	...	...	...	...	...
321761	Greenland	Denmark	2023-02-08	11971	21	11950
322050	Greenland	Denmark	2023-02-09	11971	21	11950
322339	Greenland	Denmark	2023-02-10	11971	21	11950
322628	Greenland	Denmark	2023-02-11	11971	21	11950
322917	Greenland	Denmark	2023-02-12	11971	21	11950

1118 rows × 6 columns

In [70]:
```
# Greenland
full_table.loc[full_table['Province/State']=='Greenland', 'Country/Region'] = 'Greenland
```

In [74]:
```
# filling missing values
# fill missing province/state value with ''
full_table[['Province/State']] = full_table[['Province/State']].fillna('')


# random rows
full_table.sample(6)
```

Out[74]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active
86148		Benin	2020-11-15	2844	43	2801
322599	Jiangsu	China	2023-02-11	5075	0	5075
277639	Sint Maarten	Netherlands	2022-09-08	10847	87	10760
42292		Cote d'Ivoire	2020-06-16	5679	46	5633
88786	Gansu	China	2020-11-24	181	2	179
70039		Cyprus	2020-09-20	1580	22	1558

In [75]:
```
full_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 323102 entries, 0 to 323101
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   Province/State    323102 non-null  object        
 1   Country/Region    323102 non-null  object        
 2   Date              323102 non-null  datetime64[ns]
 3   Confirmed         323102 non-null  int64         
 4   Deaths            323102 non-null  int64         
 5   Recovered/Active  323102 non-null  int64         
dtypes: datetime64[ns](1), int64(3), object(2)
memory usage: 17.3+ MB
```

In [76]:
```
#'Winter Olympics 2022' 'Summer Olympics 2020'
mask_ol_games = (full_table['Country/Region'] == 'Winter Olympics 2022') | (full_table['
olympic_games = full_table[mask_ol_games]
```

In [77]:
```
olympic_games
```

Out[77]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active
244		Summer Olympics 2020	2020-01-22	0	0	0
285		Winter Olympics 2022	2020-01-22	0	0	0
533		Summer Olympics 2020	2020-01-23	0	0	0
574		Winter Olympics 2022	2020-01-23	0	0	0
822		Summer Olympics 2020	2020-01-24	0	0	0
...	...	...	...	...	...	...
322520		Winter Olympics 2022	2023-02-10	535	0	535
322768		Summer Olympics 2020	2023-02-11	865	0	865
322809		Winter Olympics 2022	2023-02-11	535	0	535
323057		Summer Olympics 2020	2023-02-12	865	0	865
323098		Winter Olympics 2022	2023-02-12	535	0	535

2236 rows × 6 columns

In [78]:
```
full_table = full_table[~(mask_ol_games)]
```

In [79]:
```
# Ships

# ship rows containing ships with COVID-19 reported cases
ships_rows = full_table['Province/State'].str.contains('Grand Princess') | \
             full_table['Province/State'].str.contains('Diamond Princess') | \
             full_table['Country/Region'].str.contains('Diamond Princess') | \
             full_table['Country/Region'].str.contains('MS Zaandam')

ships = full_table[ships_rows]

# Latest cases from the ships
ships_latest = ships[ships['Date']==max(ships['Date'])]
# ship_latest.style.background_gradient(cmap='Pastel1_r')

# skipping rows with ships info
full_table = full_table[~(ships_rows)]
```

In [80]:
```
ships
```

Out[80]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active
42	Diamond Princess	Canada	2020-01-22	0	0	0
43	Grand Princess	Canada	2020-01-22	0	0	0
106		Diamond Princess	2020-01-22	0	0	0
175		MS Zaandam	2020-01-22	0	0	0
331	Diamond Princess	Canada	2020-01-23	0	0	0
...	...	...	...	...	...	...
322699		MS Zaandam	2023-02-11	9	2	7
322855	Diamond Princess	Canada	2023-02-12	0	1	-1
322856	Grand Princess	Canada	2023-02-12	13	0	13
322919		Diamond Princess	2023-02-12	712	13	699
322988		MS Zaandam	2023-02-12	9	2	7

4472 rows × 6 columns

In [81]:
```
full_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 316394 entries, 0 to 323101
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype         
---  ------           --------------   -----         
 0   Province/State   316394 non-null  object        
 1   Country/Region   316394 non-null  object        
 2   Date             316394 non-null  datetime64[ns]
 3   Confirmed        316394 non-null  int64         
 4   Deaths           316394 non-null  int64         
 5   Recovered/Active 316394 non-null  int64         
dtypes: datetime64[ns](1), int64(3), object(2)
memory usage: 16.9+ MB
```

In [84]:
```
full_table.isna().sum()
```

Out[84]:
```
Province/State      0
Country/Region      0
Date                0
Confirmed           0
Deaths              0
Recovered/Active    0
dtype: int64
```

In [93]:
```
full_table.to_csv('covid_19_clean_f.csv',index=False)
```

In [87]:
```
# Grouped by day, country

full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths', 'Re

# new cases
temp = full_grouped.groupby(['Country/Region', 'Date' ])['Confirmed', 'Deaths', 'Recover
temp = temp.sum().diff().reset_index()

mask = temp['Country/Region'] != temp['Country/Region'].shift(1)

temp.loc[mask, 'Confirmed'] = np.nan
temp.loc[mask, 'Deaths'] = np.nan
temp.loc[mask, 'Recovered/Active'] = np.nan

# renaming columns
temp.columns = ['Country/Region', 'Date', 'New cases', 'New deaths', 'New recovered']

# merging new values
full_grouped = pd.merge(full_grouped, temp, on=['Country/Region', 'Date'])

# filling na with 0
full_grouped = full_grouped.fillna(0)

# fixing data types
cols = ['New cases', 'New deaths', 'New recovered']
full_grouped[cols] = full_grouped[cols].astype('int')

full_grouped['New cases'] = full_grouped['New cases'].apply(lambda x: 0 if x<0 else x)

full_grouped.sample(10)
```

```
/var/folders/p9/xfvynkrd71x4qjkfsrw5xb7c0000gn/T/ipykernel_11229/896431172.py:3: FutureW
arning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be de
precated, use a list instead.
  full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths',
'Recovered/Active'].sum().reset_index()
/var/folders/p9/xfvynkrd71x4qjkfsrw5xb7c0000gn/T/ipykernel_11229/896431172.py:6: FutureW
arning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be de
precated, use a list instead.
  temp = full_grouped.groupby(['Country/Region', 'Date' ])['Confirmed', 'Deaths', 'Recov
ered/Active']
```

Out[87]:

	Date	Country/Region	Confirmed	Deaths	Recovered/Active	New cases	New deaths	New recovered
163947	2022-04-29	Andorra	41349	153	41196	0	0	0
58845	2020-11-14	Comoros	579	7	572	5	0	5
96002	2021-05-20	Sweden	1055173	14351	1040822	3411	2	3409
115010	2021-08-24	Sweden	1119358	14670	1104688	2774	2	2772

38171	2020-08-01	Serbia	25882	582	25300	330	9	321
217336	2023-01-23	North Korea	1	6	-5	0	0	0
145746	2022-01-27	Belize	49794	624	49170	735	0	735
211972	2022-12-27	Mauritania	63425	997	62428	0	0	0
1588	2020-01-30	Angola	0	0	0	0	0	0
179933	2022-07-18	Saint Vincent and the Grenadines	9183	115	9068	0	0	0

In [91]:

```
# Day wise

# table
day_wise = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered/Active',
                                        'New cases', 'New deaths', 'New recovered'].sum

# number cases per 100 cases
day_wise['Deaths / 100 Cases'] = round((day_wise['Deaths']/day_wise['Confirmed'])*100, 2
day_wise['Recovered / 100 Cases'] = round((day_wise['Recovered/Active']/day_wise['Confir
day_wise['Deaths / 100 Recovered'] = round((day_wise['Deaths']/day_wise['Recovered/Activ

# no. of countries
day_wise['No. of countries'] = full_grouped[full_grouped['Confirmed']!=0] \
                                    .groupby('Date')['Country/Region'] \
                                    .unique() \
                                    .apply(len)\
                                    .values

# fillna by 0
cols = ['Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered']
day_wise[cols] = day_wise[cols].fillna(0)

day_wise.sample(10)
```

```
/var/folders/p9/xfvynkrd71x4qjkfsrw5xb7c0000gn/T/ipykernel_11229/1316370469.py:4: Future
Warning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be d
eprecated, use a list instead.
  day_wise = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered/Active',
```

Out[91]:

	Date	Confirmed	Deaths	Recovered/Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Re
82	2020-04-13	1918859	136952	1781907	71832	6513	65319	7.14	92.86	
1072	2022-12-29	659614556	6690002	652924554	671288	2918	668370	1.01	98.99	
528	2021-07-03	183963685	4000482	179963203	373827	6903	366924	2.17	97.83	
805	2022-04-06	495612034	6199233	489412801	1219971	4323	1215648	1.25	98.75	
94	2020-04-25	2902763	223347	2679416	83205	6050	77155	7.69	92.31	
592	2021-09-05	221346170	4595994	216750176	446968	6570	440398	2.08	97.92	

490	2021-05-26	168929553	3640372	165289181	568428	12806	555622	2.15	97.85
1051	2022-12-08	647838723	6650689	641188034	809089	3007	806082	1.03	98.97
257	2020-10-05	35550788	1110778	34440010	311345	7089	304256	3.12	96.88
557	2021-08-01	198871735	4254005	194617730	483050	7419	475631	2.14	97.86

In [92]:
```
day_wise.to_csv('day_wise_f.csv', index=False)
```

In [97]:
```
# Country wise
# ============

full_grouped['Date'] = pd.to_datetime(full_grouped['Date'])

# getting latest values
country_wise = full_grouped[full_grouped['Date']==max(full_grouped['Date'])] \
                    .reset_index(drop=True) \
                    .drop('Date', axis=1)

print(country_wise.shape)

# group by country
country_wise = country_wise.groupby('Country/Region')['Confirmed', 'Deaths',
                                                      'Recovered/Active',
                                                      'New cases', 'New deaths', 'New re
print(country_wise.shape)


# per 100 cases
country_wise['Deaths / 100 Cases'] = round((country_wise['Deaths']/country_wise['Confirm
country_wise['Recovered / 100 Cases'] = round((country_wise['Recovered/Active']/country_
country_wise['Deaths / 100 Recovered'] = round((country_wise['Deaths']/country_wise['Rec

cols = ['Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered']
country_wise[cols] = country_wise[cols].fillna(0)




country_wise.tail()
```

```
(198, 7)
(198, 7)
```

```
/var/folders/p9/xfvynkrd71x4qjkfsrw5xb7c0000gn/T/ipykernel_11229/2286068760.py:14: Futur
eWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be
deprecated, use a list instead.
  country_wise = country_wise.groupby('Country/Region')['Confirmed', 'Deaths',
```

Out[97]:

	Country/Region	Confirmed	Deaths	Recovered/Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases
193	Vietnam	11526692	43186	11483506	41	0	41	0.37	99.63
194	West Bank and Gaza	703228	5708	697520	0	0	0	0.81	99.19
195	Yemen	11945	2159	9786	0	0	0	18.07	81.93
196	Zambia	342288	4051	338237	0	0	0	1.18	98.82
197	Zimbabwe	263083	5659	257424	0	0	0	2.15	97.85

In [98]:
```
country_wise.to_csv('country_wise_f.csv', index=False)
```

In [101…
```
who_region = {}

# African Region AFRO
afro = "Algeria, Angola, Cabo Verde, Eswatini, Sao Tome and Principe, Benin, South Sudan
afro = [i.strip() for i in afro.split(',')]
for i in afro:
    who_region[i] = 'Africa'

# Region of the Americas PAHO
paho = 'Antigua and Barbuda, Argentina, Bahamas, Barbados, Belize, Bolivia, Brazil, Cana
paho = [i.strip() for i in paho.split(',')]
for i in paho:
    who_region[i] = 'Americas'

# South-East Asia Region SEARO
searo = 'Bangladesh, Bhutan, North Korea, India, Indonesia, Maldives, Myanmar, Burma, Ne
searo = [i.strip() for i in searo.split(',')]
for i in searo:
    who_region[i] = 'South-East Asia'

# European Region EURO
euro = 'Albania, Andorra, Greenland, Kosovo, Holy See, Liechtenstein, Armenia, Czechia,
euro = [i.strip() for i in euro.split(',')]
for i in euro:
    who_region[i] = 'Europe'

# Eastern Mediterranean Region EMRO
emro = 'Afghanistan, Bahrain, Djibouti, Egypt, Iran, Iraq, Jordan, Kuwait, Lebanon, Liby
emro = [i.strip() for i in emro.split(',')]
for i in emro:
    who_region[i] = 'Eastern Mediterranean'

# Western Pacific Region WPRO
wpro = 'Australia, Brunei, Cambodia, China, Cook Islands, Fiji, Japan, Kiribati, Laos, M
wpro = [i.strip() for i in wpro.split(',')]
for i in wpro:
    who_region[i] = 'Western Pacific'
```

In [102…
```
# add 'WHO Region' column
full_table['WHO Region'] = full_table['Country/Region'].map(who_region)

# find missing values
full_table[full_table['WHO Region'].isna()]['Country/Region'].unique()
```

Out[102]:
```
array(['Antarctica'], dtype=object)
```

In [112…
```
full_table.fillna('Antarctica', inplace=True)
```

In [113…
```
full_table[full_table['WHO Region'].isna()]['Country/Region'].unique()
```

Out[113]:
```
array([], dtype=object)
```

In [103…
```
full_table.sample(10)
```

Out[103]:

	Province/State	Country/Region	Date	Confirmed	Deaths	Recovered/Active	WHO Region
30421	Jiangxi	China	2020-05-06	937	1	936	Western Pacific
106034		US	2021-01-	25010536	419237	24591299	Americas

			22				
225846		Greece	2022-03-13	2635614	26562	2609052	Europe
178084	Beijing	China	2021-09-29	1123	9	1114	Western Pacific
20352	Guadeloupe	France	2020-04-01	125	6	119	Europe
161508		Sweden	2021-08-02	1100040	14655	1085385	Europe
155573	Yunnan	China	2021-07-13	537	2	535	Western Pacific
173640		South Sudan	2021-09-13	11623	120	11503	Africa
91107	Hubei	China	2020-12-02	68149	4512	63637	Western Pacific
50616	British Columbia	Canada	2020-07-15	3149	189	2960	Americas

In [114…
```
full_table.to_csv('covid_who.csv')
```

In [100…
```
countries = list(country_wise['Country/Region'].unique())
countries
```

Out[100]:
```
['Afghanistan',
 'Albania',
 'Algeria',
 'Andorra',
 'Angola',
 'Antarctica',
 'Antigua and Barbuda',
 'Argentina',
 'Armenia',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas',
 'Bahrain',
 'Bangladesh',
 'Barbados',
 'Belarus',
 'Belgium',
 'Belize',
 'Benin',
 'Bhutan',
 'Bolivia',
 'Bosnia and Herzegovina',
 'Botswana',
 'Brazil',
 'Brunei',
 'Bulgaria',
 'Burkina Faso',
 'Burma',
 'Burundi',
 'Cabo Verde',
 'Cambodia',
 'Cameroon',
 'Canada',
 'Central African Republic',
 'Chad',
```

```
'Chile',
'China',
'Colombia',
'Comoros',
'Congo (Brazzaville)',
'Congo (Kinshasa)',
'Costa Rica',
"Cote d'Ivoire",
'Croatia',
'Cuba',
'Cyprus',
'Czechia',
'Denmark',
'Djibouti',
'Dominica',
'Dominican Republic',
'Ecuador',
'Egypt',
'El Salvador',
'Equatorial Guinea',
'Eritrea',
'Estonia',
'Eswatini',
'Ethiopia',
'Fiji',
'Finland',
'France',
'Gabon',
'Gambia',
'Georgia',
'Germany',
'Ghana',
'Greece',
'Greenland',
'Grenada',
'Guatemala',
'Guinea',
'Guinea-Bissau',
'Guyana',
'Haiti',
'Holy See',
'Honduras',
'Hungary',
'Iceland',
'India',
'Indonesia',
'Iran',
'Iraq',
'Ireland',
'Israel',
'Italy',
'Jamaica',
'Japan',
'Jordan',
'Kazakhstan',
'Kenya',
'Kiribati',
'Kosovo',
'Kuwait',
'Kyrgyzstan',
'Laos',
'Latvia',
'Lebanon',
'Lesotho',
'Liberia',
'Libya',
```

'Liechtenstein',
'Lithuania',
'Luxembourg',
'Madagascar',
'Malawi',
'Malaysia',
'Maldives',
'Mali',
'Malta',
'Marshall Islands',
'Mauritania',
'Mauritius',
'Mexico',
'Micronesia',
'Moldova',
'Monaco',
'Mongolia',
'Montenegro',
'Morocco',
'Mozambique',
'Namibia',
'Nauru',
'Nepal',
'Netherlands',
'New Zealand',
'Nicaragua',
'Niger',
'Nigeria',
'North Korea',
'North Macedonia',
'Norway',
'Oman',
'Pakistan',
'Palau',
'Panama',
'Papua New Guinea',
'Paraguay',
'Peru',
'Philippines',
'Poland',
'Portugal',
'Qatar',
'Romania',
'Russia',
'Rwanda',
'Saint Kitts and Nevis',
'Saint Lucia',
'Saint Vincent and the Grenadines',
'Samoa',
'San Marino',
'Sao Tome and Principe',
'Saudi Arabia',
'Senegal',
'Serbia',
'Seychelles',
'Sierra Leone',
'Singapore',
'Slovakia',
'Slovenia',
'Solomon Islands',
'Somalia',
'South Africa',
'South Korea',
'South Sudan',
'Spain',
'Sri Lanka',

```
 'Sudan',
 'Suriname',
 'Sweden',
 'Switzerland',
 'Syria',
 'Taiwan*',
 'Tajikistan',
 'Tanzania',
 'Thailand',
 'Timor-Leste',
 'Togo',
 'Tonga',
 'Trinidad and Tobago',
 'Tunisia',
 'Turkey',
 'Tuvalu',
 'US',
 'Uganda',
 'Ukraine',
 'United Arab Emirates',
 'United Kingdom',
 'Uruguay',
 'Uzbekistan',
 'Vanuatu',
 'Venezuela',
 'Vietnam',
 'West Bank and Gaza',
 'Yemen',
 'Zambia',
 'Zimbabwe']
```

In [132…
```
#importing gdp dataset
gdp_df = pd.read_csv('gdp.csv', skiprows=3, usecols=['Country Name', '2019', '2020', '20
gdp_df.head()
```

Out[132]:

	Country Name	2019	2020	2021
0	Aruba	3.368970e+09	2.610039e+09	3.126019e+09
1	Africa Eastern and Southern	1.001017e+12	9.274845e+11	1.080712e+12
2	Afghanistan	1.890449e+10	2.014344e+10	1.478686e+10
3	Africa Western and Central	7.947191e+11	7.847997e+11	8.401873e+11
4	Angola	6.930911e+10	5.361907e+10	6.740429e+10

In [133…
```
gdp_df.columns = ['Country', '2019', '2020', '2021'] #changing column name
```

In [124…
```
#!pip install fuzzywuzzy
# !pip install python-levenshtein
```

```
Collecting python-levenshtein
  Downloading python_Levenshtein-0.20.9-py3-none-any.whl (9.4 kB)
Collecting Levenshtein==0.20.9
  Downloading Levenshtein-0.20.9-cp39-cp39-macosx_10_9_x86_64.whl (131 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 131.4/131.4 kB 853.4 kB/s eta 0:00:00a 0:00:0
1
Collecting rapidfuzz<3.0.0,>=2.3.0
  Downloading rapidfuzz-2.13.7-cp39-cp39-macosx_10_9_x86_64.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 4.8 MB/s eta 0:00:0000:0100:01
Installing collected packages: rapidfuzz, Levenshtein, python-levenshtein
Successfully installed Levenshtein-0.20.9 python-levenshtein-0.20.9 rapidfuzz-2.13.7
```

In [128…
```
from fuzzywuzzy import fuzz, process
```

In [134…
```
#string comparing to change the countries name to match covid dataset

for country in countries:
    matches=process.extract(country,gdp_df['Country'],limit=gdp_df.shape[0])
    for potential_match in matches:
        if potential_match[1]>=80:
            gdp_df.loc[gdp_df['Country']==potential_match[0], 'Country'] = country
```

In [135…
```
gdp_df['Country'].unique()
```

Out[135]:
```
array(['Aruba', 'South Africa', 'Afghanistan', 'Angola', 'Albania',
       'Andorra', 'United Arab Emirates', 'Argentina', 'Armenia', 'Samoa',
       'Sao Tome and Principe', 'US', 'Azerbaijan', 'Burundi', 'Belgium',
       'Benin', 'Burkina Faso', 'Bangladesh', 'Bulgaria', 'Bahrain',
       'Bahamas', 'Bosnia and Herzegovina', 'Belize', 'Bermuda',
       'Bolivia', 'Brazil', 'Barbados', 'Brunei', 'Bhutan', 'Botswana',
       'Central African Republic', 'Canada', 'Switzerland',
       'Channel Islands', 'Chile', 'China', "Cote d'Ivoire", 'Cameroon',
       'Congo, Dem. Rep.', 'Congo (Brazzaville)', 'Colombia', 'Comoros',
       'Cabo Verde', 'Costa Rica', 'Caribbean small states', 'Cuba',
       'Curacao', 'Cayman Islands', 'Czechia', 'Germany', 'Djibouti',
       'Dominican Republic', 'Denmark', 'Algeria',
       'East Asia & Pacific (excluding high income)',
       'Early-demographic dividend', 'East Asia & Pacific',
       'Europe & Central Asia', 'Ecuador', 'Egypt', 'Euro area',
       'Eritrea', 'Spain', 'Estonia', 'Ethiopia', 'European Union',
       'Finland', 'Fiji', 'France', 'Faroe Islands', 'Micronesia',
       'Gabon', 'United Kingdom', 'Georgia', 'Ghana', 'Gibraltar',
       'Guinea-Bissau', 'Zambia', 'Greece', 'Grenada', 'Greenland',
       'Malta', 'Guam', 'Guyana', 'High income', 'Honduras',
       'Heavily indebted poor countries (HIPC)', 'Croatia', 'Haiti',
       'Hungary', 'IBRD only', 'IDA & IBRD total', 'IDA total',
       'IDA blend', 'Indonesia', 'IDA only', 'Isle of Man', 'India',
       'Not classified', 'Ireland', 'Iran', 'Iraq', 'Israel', 'Italy',
       'Jamaica', 'Jordan', 'Japan', 'Kazakhstan', 'Kenya', 'Cambodia',
       'Kiribati', 'Korea, Rep.', 'Kuwait',
       'Latin America & Caribbean (excluding high income)', 'Lao PDR',
       'Lebanon', 'Liberia', 'Libya', 'Latin America & Caribbean',
       'Least developed countries: UN classification', 'Low income',
       'Liechtenstein', 'Sri Lanka', 'Lower middle income',
       'Low & middle income', 'Lesotho', 'Late-demographic dividend',
       'Lithuania', 'Luxembourg', 'Latvia', 'St. Martin (French part)',
       'Morocco', 'Monaco', 'Moldova', 'Madagascar', 'Maldives',
       'South Korea', 'Mexico', 'Marshall Islands', 'Middle income',
       'North Macedonia', 'Somalia', 'Myanmar', 'Montenegro', 'Mongolia',
       'Mozambique', 'Mauritania', 'Malaysia', 'North America', 'Namibia',
       'New Caledonia', 'Nigeria', 'Nicaragua', 'Netherlands', 'Norway',
       'Nepal', 'Nauru', 'New Zealand', 'OECD members', 'Romania',
       'Other small states', 'Pakistan', 'Panama', 'Peru', 'Philippines',
       'Palau', 'Poland', 'Pre-demographic dividend', 'Puerto Rico',
       'Portugal', 'Paraguay', 'Pacific island small states',
       'Post-demographic dividend', 'French Polynesia', 'Qatar', 'Rwanda',
       'Saudi Arabia', 'Sudan', 'Senegal', 'Singapore', 'Solomon Islands',
       'Sierra Leone', 'El Salvador', 'San Marino', 'Serbia',
       'Small states', 'Suriname', 'Slovenia', 'Sweden', 'Eswatini',
       'Sint Maarten (Dutch part)', 'Seychelles', 'Syria', 'Chad',
       'East Asia & Pacific (IDA & IBRD countries)', 'Togo', 'Thailand',
       'Tajikistan', 'Turkmenistan', 'Timor-Leste', 'Tonga', 'Tunisia',
       'Turkiye', 'Tuvalu', 'Tanzania', 'Uganda', 'Ukraine',
       'Upper middle income', 'Uruguay', 'Uzbekistan', 'Venezuela',
       'British Virgin Islands', 'Virgin Islands (U.S.)', 'Vietnam',
       'Vanuatu', 'World', 'Kosovo', 'Yemen', 'Zimbabwe'], dtype=object)
```

In [141…
```
country_gdp = pd.merge(country_wise, gdp_df, left_on='Country/Region', right_on='Country
country_gdp.columns
```

Out[141]:
```
Index(['Country/Region', 'Confirmed', 'Deaths', 'Recovered/Active',
       'New cases', 'New deaths', 'New recovered', 'Deaths / 100 Cases',
       'Recovered / 100 Cases', 'Deaths / 100 Recovered', 'Country', '2019',
       '2020', '2021'],
      dtype='object')
```

In [146…
```
country_gdp = country_gdp[['Country/Region', 'Confirmed',  '2019','2020', '2021']]
country_gdp.dropna(inplace=True)
country_gdp
```

Out[146]:

	Country/Region	Confirmed	2019	2020	2021
0	Afghanistan	208982	1.890449e+10	2.014344e+10	1.478686e+10
1	Albania	334255	1.540183e+10	1.513187e+10	1.825579e+10
2	Algeria	271409	1.717674e+11	1.450092e+11	1.630444e+11
3	Andorra	47860	3.155065e+09	2.891022e+09	3.330282e+09
4	Angola	105184	6.930911e+10	5.361907e+10	6.740429e+10
…	…	…	…	…	…
237	Vanuatu	12014	9.365263e+08	8.968799e+08	9.563327e+08
239	Vietnam	11526692	3.343653e+11	3.466158e+11	3.661376e+11
242	Zambia	342288	1.813608e+09	1.812169e+09	2.038417e+09
243	Zambia	342288	2.330867e+10	1.811063e+10	2.214763e+10
244	Zimbabwe	263083	2.183223e+10	2.150970e+10	2.837124e+10

199 rows × 5 columns

In [149…
```
#function to convert gdp into billions
def gdp_billion(x):
    return x/1000_000_000
```

In [154…
```
col_to_bil = ['2019', '2020', '2021']
gdp_col = country_gdp[col_to_bil]
gdp_col = gdp_col.apply(gdp_billion)
country_gdp[col_to_bil] = gdp_col
country_gdp.columns = ['Country', 'Confirmed', '2019GDP/billions', '2020GDP/billion' , '
country_gdp
```

Out[154]:

	Country	Confirmed	2019GDP/billions	2020GDP/billion	2019GDP/billion
0	Afghanistan	208982	1.890449e-08	2.014344e-08	1.478686e-08
1	Albania	334255	1.540183e-08	1.513187e-08	1.825579e-08
2	Algeria	271409	1.717674e-07	1.450092e-07	1.630444e-07
3	Andorra	47860	3.155065e-09	2.891022e-09	3.330282e-09
4	Angola	105184	6.930911e-08	5.361907e-08	6.740429e-08
…	…	…	…	…	…
237	Vanuatu	12014	9.365263e-10	8.968799e-10	9.563327e-10
239	Vietnam	11526692	3.343653e-07	3.466158e-07	3.661376e-07
242	Zambia	342288	1.813608e-09	1.812169e-09	2.038417e-09
243	Zambia	342288	2.330867e-08	1.811063e-08	2.214763e-08
244	Zimbabwe	263083	2.183223e-08	2.150970e-08	2.837124e-08

199 rows × 5 columns

In [155…
```
country_gdp.to_csv('gdp19to21.csv')
```

In [ ]:
```
```