# Covid_19_Trend_Analysis_by_B_Group_No_12

April 16, 2025

## 1 Analysing COVID-19 Data by Div.B Group No.12

### 1.1 16010122203 - Anish Talari

### 1.2 16010122205 - Krish Telang

### 1.3 16010122214 - Arya Torne

### 1.4 16010122230 - Aryan Puri

### 1.5 16010122232 - Mansi Sharma

### 1.6 Data Preparation

1. Filtering and Selecting
2. Treating missing values
3. Removing Duplicates
4. Concatening and Transforming
5. Group and Aggregation

### 1.7 Installing pandas

[1]:
```
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in ./.local/lib/python3.10/site-packages
(2.2.3)
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3/dist-packages
(from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
./.local/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: numpy>=1.22.4 in ./.local/lib/python3.10/site-
packages (from pandas) (2.2.4)
Requirement already satisfied: tzdata>=2022.7 in ./.local/lib/python3.10/site-
packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from
python-dateutil>=2.8.2->pandas) (1.16.0)
```

### 1.8 Filtering and Selecting

[2]:
```
import pandas as pd
```

### 1.9 Data Loading

[3]:
```
covid_19_data_url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/
 ↪refs/heads/master/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_confirmed_global.csv'
covid_19 = pd.read_csv(covid_19_data_url)
covid_19
```

[3]:
```
    Province/State        Country/Region        Lat        Long  1/22/20  \
0              NaN           Afghanistan  33.939110   67.709953        0
1              NaN               Albania  41.153300   20.168300        0
2              NaN               Algeria  28.033900    1.659600        0
3              NaN               Andorra  42.506300    1.521800        0
4              NaN                Angola -11.202700   17.873900        0
..             …                     …         …          …      …
284            NaN  West Bank and Gaza  31.952200   35.233200        0
285            NaN  Winter Olympics 2022  39.904200  116.407400        0
286            NaN                 Yemen  15.552727   48.516388        0
287            NaN                Zambia -13.133897   27.849332        0
288            NaN              Zimbabwe -19.015438   29.154857        0

     1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  …  2/28/23  3/1/23  \
0          0        0        0        0        0  …   209322  209340
1          0        0        0        0        0  …   334391  334408
2          0        0        0        0        0  …   271441  271448
3          0        0        0        0        0  …    47866   47875
4          0        0        0        0        0  …   105255  105277
..       …        …        …        …        …  …        …       …
284        0        0        0        0        0  …   703228  703228
285        0        0        0        0        0  …      535     535
286        0        0        0        0        0  …    11945   11945
287        0        0        0        0        0  …   343012  343012
288        0        0        0        0        0  …   263921  264127

     3/2/23  3/3/23  3/4/23  3/5/23  3/6/23  3/7/23  3/8/23  3/9/23
0    209358  209362  209369  209390  209406  209436  209451  209451
1    334408  334427  334427  334427  334427  334427  334443  334457
2    271463  271469  271469  271477  271477  271490  271494  271496
3     47875   47875   47875   47875   47875   47875   47890   47890
4    105277  105277  105277  105277  105277  105277  105288  105288
..      …       …       …       …       …       …       …       …
284  703228  703228  703228  703228  703228  703228  703228  703228
285     535     535     535     535     535     535     535     535
```

```
286   11945   11945   11945   11945   11945   11945   11945   11945
287  343079  343079  343079  343135  343135  343135  343135  343135
288  264127  264127  264127  264127  264127  264127  264276  264276

[289 rows x 1147 columns]
```

[19]:
```
covid_19_data_url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/
 ↪refs/heads/master/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_confirmed_global.csv'
covid_19 = pd.read_csv(covid_19_data_url)
covid_19.head()
```

[19]:
```
  Province/State Country/Region       Lat       Long  1/22/20  1/23/20  \
0            NaN    Afghanistan  33.93911  67.709953        0        0
1            NaN        Albania  41.15330  20.168300        0        0
2            NaN        Algeria  28.03390   1.659600        0        0
3            NaN        Andorra  42.50630   1.521800        0        0
4            NaN         Angola -11.20270  17.873900        0        0

   1/24/20  1/25/20  1/26/20  1/27/20  …  2/28/23  3/1/23  3/2/23  3/3/23  \
0        0        0        0        0  …   209322  209340  209358  209362
1        0        0        0        0  …   334391  334408  334408  334427
2        0        0        0        0  …   271441  271448  271463  271469
3        0        0        0        0  …    47866   47875   47875   47875
4        0        0        0        0  …   105255  105277  105277  105277

   3/4/23  3/5/23  3/6/23  3/7/23  3/8/23  3/9/23
0  209369  209390  209406  209436  209451  209451
1  334427  334427  334427  334427  334443  334457
2  271469  271477  271477  271490  271494  271496
3   47875   47875   47875   47875   47890   47890
4  105277  105277  105277  105277  105288  105288

[5 rows x 1147 columns]
```

### 1.10 Selecting one field

[20]:
```
covid_lat = covid_19['Lat']
covid_lat
```

[20]:
```
0      33.939110
1      41.153300
2      28.033900
3      42.506300
4     -11.202700
         …
284    31.952200
```

```
285    39.904200
286    15.552727
287   -13.133897
288   -19.015438
Name: Lat, Length: 289, dtype: float64
```

### 1.11 Selecting multiple fields

[4]:
```
covid_lat_long = covid_19[['Lat','Long']]
covid_lat_long
```

[4]:
```
            Lat        Long
0     33.939110   67.709953
1     41.153300   20.168300
2     28.033900    1.659600
3     42.506300    1.521800
4    -11.202700   17.873900
..          …          …
284   31.952200   35.233200
285   39.904200  116.407400
286   15.552727   48.516388
287  -13.133897   27.849332
288  -19.015438   29.154857

[289 rows x 2 columns]
```

### 1.12 Selecting specific fields with specific rows

[5]:
```
covid_se = covid_19.loc[[1,4],['Lat','Long']]
covid_se
```

[5]:
```
        Lat     Long
1  41.1533  20.1683
4 -11.2027  17.8739
```

### 1.13 Selecting a range of fields with specific rows

[6]:
```
covid_range = covid_19.loc[1:4,['Lat','Long']]
covid_range
```

[6]:
```
        Lat     Long
1  41.1533  20.1683
2  28.0339   1.6596
3  42.5063   1.5218
4 -11.2027  17.8739
```

### 1.14 For Latitude greater then 31

[7]:
```
covid_filter = covid_19[covid_19['Lat']>31]
covid_filter
```

[7]:
```
    Province/State        Country/Region        Lat        Long  1/22/20  \
0              NaN           Afghanistan  33.939110   67.709953        0
1              NaN               Albania  41.153300   20.168300        0
3              NaN               Andorra  42.506300    1.521800        0
8              NaN               Armenia  40.069100   45.038200        0
17             NaN               Austria  47.516200   14.550100        0
..             …                     …         …          …      …
273         Jersey        United Kingdom  49.213800   -2.135800        0
278            NaN        United Kingdom  55.378100   -3.436000        0
280            NaN            Uzbekistan  41.377491   64.585262        0
284            NaN    West Bank and Gaza  31.952200   35.233200        0
285            NaN  Winter Olympics 2022  39.904200  116.407400        0

     1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  …   2/28/23    3/1/23  \
0          0        0        0        0        0  …    209322    209340
1          0        0        0        0        0  …    334391    334408
3          0        0        0        0        0  …     47866     47875
8          0        0        0        0        0  …    446819    446819
17         0        0        0        0        0  …   5911294   5919616
..       …      …      …      …      …  …       …         …
273        0        0        0        0        0  …     66391     66391
278        0        0        0        0        0  …  24370150  24370150
280        0        0        0        0        0  …    250932    251071
284        0        0        0        0        0  …    703228    703228
285        0        0        0        0        0  …       535       535

       3/2/23    3/3/23    3/4/23    3/5/23    3/6/23    3/7/23    3/8/23  \
0      209358    209362    209369    209390    209406    209436    209451
1      334408    334427    334427    334427    334427    334427    334443
3       47875     47875     47875     47875     47875     47875     47890
8      446819    446819    446819    446819    446819    446819    447308
17    5926148   5931247   5936666   5940935   5943417   5949418   5955860
..        …         …         …         …         …         …
273     66391     66391     66391     66391     66391     66391     66391
278  24396530  24396530  24396530  24396530  24396530  24396530  24396530
280    251071    251071    251071    251071    251071    251071    251247
284    703228    703228    703228    703228    703228    703228    703228
285       535       535       535       535       535       535       535

       3/9/23
0      209451
1      334457
```

```
3       47890
8      447308
17    5961143
..        …
273     66391
278  24425309
280    251247
284    703228
285       535

[112 rows x 1147 columns]
```

### 1.15 Dropping Columns

[8]:
```
covid_drop_main = covid_19.drop(['Lat','Long'],axis = 1)
covid_drop_main
```

[8]:
```
    Province/State        Country/Region  1/22/20  1/23/20  1/24/20  1/25/20  \
0              NaN           Afghanistan        0        0        0        0
1              NaN               Albania        0        0        0        0
2              NaN               Algeria        0        0        0        0
3              NaN               Andorra        0        0        0        0
4              NaN                Angola        0        0        0        0
..             …                     …      …       …        …        …
284            NaN   West Bank and Gaza        0        0        0        0
285            NaN  Winter Olympics 2022        0        0        0        0
286            NaN                 Yemen        0        0        0        0
287            NaN                Zambia        0        0        0        0
288            NaN              Zimbabwe        0        0        0        0

     1/26/20  1/27/20  1/28/20  1/29/20  …  2/28/23  3/1/23  3/2/23  3/3/23  \
0          0        0        0        0  …   209322  209340  209358  209362
1          0        0        0        0  …   334391  334408  334408  334427
2          0        0        0        0  …   271441  271448  271463  271469
3          0        0        0        0  …    47866   47875   47875   47875
4          0        0        0        0  …   105255  105277  105277  105277
..       …        …        …        …  …        …       …       …
284        0        0        0        0  …   703228  703228  703228  703228
285        0        0        0        0  …      535     535     535     535
286        0        0        0        0  …    11945   11945   11945   11945
287        0        0        0        0  …   343012  343012  343079  343079
288        0        0        0        0  …   263921  264127  264127  264127

     3/4/23  3/5/23  3/6/23  3/7/23  3/8/23  3/9/23
0    209369  209390  209406  209436  209451  209451
1    334427  334427  334427  334427  334443  334457
2    271469  271477  271477  271490  271494  271496
```

```
3     47875   47875   47875   47875   47890   47890
4    105277  105277  105277  105277  105288  105288
..      …       …       …       …       …       …
284  703228  703228  703228  703228  703228  703228
285     535     535     535     535     535     535
286   11945   11945   11945   11945   11945   11945
287  343079  343135  343135  343135  343135  343135
288  264127  264127  264127  264127  264276  264276

[289 rows x 1145 columns]
```

### 1.16 Treating missing data

By default the missing value are represented as NaN,"Not a Number", If the dataset has 0s, 99s or 999s, be sure to drop or approximate them as u would with missing values

### 1.17 Counting missing values

[9]:
```
covid_19.isnull().sum()
```

[9]:
```
Province/State    198
Country/Region      0
Lat                 2
Long                2
1/22/20             0
                   …
3/5/23              0
3/6/23              0
3/7/23              0
3/8/23              0
3/9/23              0
Length: 1147, dtype: int64
```

### 1.18 Filling null values

[10]:
```
covid_fill_up = covid_19.fillna("Not available")
covid_fill_up
```

[10]:
```
    Province/State        Country/Region        Lat       Long  1/22/20  \
0    Not available           Afghanistan   33.93911  67.709953        0
1    Not available               Albania    41.1533    20.1683        0
2    Not available               Algeria    28.0339     1.6596        0
3    Not available               Andorra    42.5063     1.5218        0
4    Not available                Angola   -11.2027    17.8739        0
..             …                     …          …         …       …
284  Not available  West Bank and Gaza    31.9522    35.2332        0
285  Not available  Winter Olympics 2022    39.9042   116.4074        0
```

```
286  Not available                Yemen  15.552727  48.516388        0
287  Not available               Zambia -13.133897  27.849332        0
288  Not available             Zimbabwe -19.015438  29.154857        0

     1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  …  2/28/23  3/1/23  \
0          0        0        0        0        0  …   209322  209340
1          0        0        0        0        0  …   334391  334408
2          0        0        0        0        0  …   271441  271448
3          0        0        0        0        0  …    47866   47875
4          0        0        0        0        0  …   105255  105277
..       …        …        …        …        …  …        …       …
284        0        0        0        0        0  …   703228  703228
285        0        0        0        0        0  …      535     535
286        0        0        0        0        0  …    11945   11945
287        0        0        0        0        0  …   343012  343012
288        0        0        0        0        0  …   263921  264127

     3/2/23  3/3/23  3/4/23  3/5/23  3/6/23  3/7/23  3/8/23  3/9/23
0    209358  209362  209369  209390  209406  209436  209451  209451
1    334408  334427  334427  334427  334427  334427  334443  334457
2    271463  271469  271469  271477  271477  271490  271494  271496
3     47875   47875   47875   47875   47875   47875   47890   47890
4    105277  105277  105277  105277  105277  105277  105288  105288
..      …       …       …       …       …       …       …       …
284  703228  703228  703228  703228  703228  703228  703228  703228
285     535     535     535     535     535     535     535     535
286   11945   11945   11945   11945   11945   11945   11945   11945
287  343079  343079  343079  343135  343135  343135  343135  343135
288  264127  264127  264127  264127  264127  264127  264276  264276

[289 rows x 1147 columns]
```

[11]:
```
covid_drop = covid_19.dropna()
covid_drop
```

[11]:
```
                                  Province/State  Country/Region        Lat  \
9                     Australian Capital Territory       Australia -35.473500
10                                New South Wales       Australia -33.868800
11                             Northern Territory       Australia -12.463400
12                                     Queensland       Australia -27.469800
13                                South Australia       Australia -34.928500
..                                            …               …        …
273                                       Jersey  United Kingdom  49.213800
274                                   Montserrat  United Kingdom  16.742498
275                               Pitcairn Islands  United Kingdom -24.376800
276  Saint Helena, Ascension and Tristan da Cunha  United Kingdom  -7.946700
277                       Turks and Caicos Islands  United Kingdom  21.694000
```

```
           Long  1/22/20  1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  …  \
9    149.012400        0        0        0        0        0        0  …
10   151.209300        0        0        0        0        3        4  …
11   130.845600        0        0        0        0        0        0  …
12   153.025100        0        0        0        0        0        0  …
13   138.600700        0        0        0        0        0        0  …
..          …      …      …      …      …      …      …  …
273   -2.135800        0        0        0        0        0        0  …
274  -62.187366        0        0        0        0        0        0  …
275 -128.324200        0        0        0        0        0        0  …
276  -14.355900        0        0        0        0        0        0  …
277  -71.797900        0        0        0        0        0        0  …

     2/28/23   3/1/23   3/2/23   3/3/23   3/4/23   3/5/23   3/6/23   3/7/23  \
9     232018   232018   232619   232619   232619   232619   232619   232619
10   3900969  3900969  3908129  3908129  3908129  3908129  3908129  3908129
11    104931   104931   105021   105021   105021   105021   105021   105021
12   1796633  1796633  1800236  1800236  1800236  1800236  1800236  1800236
13    880207   880207   881911   881911   881911   881911   881911   881911
..       …        …        …        …        …        …        …        …
273    66391    66391    66391    66391    66391    66391    66391    66391
274     1403     1403     1403     1403     1403     1403     1403     1403
275        4        4        4        4        4        4        4        4
276     2166     2166     2166     2166     2166     2166     2166     2166
277     6551     6551     6551     6551     6551     6551     6551     6557

      3/8/23   3/9/23
9     232619   232974
10   3908129  3915992
11    105021   105111
12   1800236  1800236
13    881911   883620
..       …        …
273    66391    66391
274     1403     1403
275        4        4
276     2166     2166
277     6557     6561

[89 rows x 1147 columns]
```

### 1.19 Removing Duplicate Data

To remove redundanct or incorrect results

[12]:
```
covid_drop_dup = covid_19.drop_duplicates()

covid_drop_dup
```

[12]:

| | Province/State | Country/Region | Lat | Long | 1/22/20 | \ |
|---|---|---|---|---|---|---|
| 0 | NaN | Afghanistan | 33.939110 | 67.709953 | 0 | |
| 1 | NaN | Albania | 41.153300 | 20.168300 | 0 | |
| 2 | NaN | Algeria | 28.033900 | 1.659600 | 0 | |
| 3 | NaN | Andorra | 42.506300 | 1.521800 | 0 | |
| 4 | NaN | Angola | -11.202700 | 17.873900 | 0 | |
| .. | … | … | … | … | … | |
| 284 | NaN | West Bank and Gaza | 31.952200 | 35.233200 | 0 | |
| 285 | NaN | Winter Olympics 2022 | 39.904200 | 116.407400 | 0 | |
| 286 | NaN | Yemen | 15.552727 | 48.516388 | 0 | |
| 287 | NaN | Zambia | -13.133897 | 27.849332 | 0 | |
| 288 | NaN | Zimbabwe | -19.015438 | 29.154857 | 0 | |

| | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 | 1/27/20 | … | 2/28/23 | 3/1/23 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | … | 209322 | 209340 | |
| 1 | 0 | 0 | 0 | 0 | 0 | … | 334391 | 334408 | |
| 2 | 0 | 0 | 0 | 0 | 0 | … | 271441 | 271448 | |
| 3 | 0 | 0 | 0 | 0 | 0 | … | 47866 | 47875 | |
| 4 | 0 | 0 | 0 | 0 | 0 | … | 105255 | 105277 | |
| .. | … | … | … | … | … | … | … | … | |
| 284 | 0 | 0 | 0 | 0 | 0 | … | 703228 | 703228 | |
| 285 | 0 | 0 | 0 | 0 | 0 | … | 535 | 535 | |
| 286 | 0 | 0 | 0 | 0 | 0 | … | 11945 | 11945 | |
| 287 | 0 | 0 | 0 | 0 | 0 | … | 343012 | 343012 | |
| 288 | 0 | 0 | 0 | 0 | 0 | … | 263921 | 264127 | |

| | 3/2/23 | 3/3/23 | 3/4/23 | 3/5/23 | 3/6/23 | 3/7/23 | 3/8/23 | 3/9/23 |
|---|---|---|---|---|---|---|---|---|
| 0 | 209358 | 209362 | 209369 | 209390 | 209406 | 209436 | 209451 | 209451 |
| 1 | 334408 | 334427 | 334427 | 334427 | 334427 | 334427 | 334443 | 334457 |
| 2 | 271463 | 271469 | 271469 | 271477 | 271477 | 271490 | 271494 | 271496 |
| 3 | 47875 | 47875 | 47875 | 47875 | 47875 | 47875 | 47890 | 47890 |
| 4 | 105277 | 105277 | 105277 | 105277 | 105277 | 105277 | 105288 | 105288 |
| .. | … | … | … | … | … | … | … | |
| 284 | 703228 | 703228 | 703228 | 703228 | 703228 | 703228 | 703228 | 703228 |
| 285 | 535 | 535 | 535 | 535 | 535 | 535 | 535 | 535 |
| 286 | 11945 | 11945 | 11945 | 11945 | 11945 | 11945 | 11945 | 11945 |
| 287 | 343079 | 343079 | 343079 | 343135 | 343135 | 343135 | 343135 | 343135 |
| 288 | 264127 | 264127 | 264127 | 264127 | 264127 | 264127 | 264276 | 264276 |

[289 rows x 1147 columns]

### 1.20 Drop Duplicates from a specific column

[13]:
```
covid_drop_col=covid_19.drop_duplicates(['Country/Region'])
covid_drop_col
```

[13]:
```
    Province/State        Country/Region        Lat        Long  1/22/20  \
0              NaN           Afghanistan  33.939110   67.709953        0
1              NaN               Albania  41.153300   20.168300        0
2              NaN               Algeria  28.033900    1.659600        0
3              NaN               Andorra  42.506300    1.521800        0
4              NaN                Angola -11.202700   17.873900        0
..             …                     …         …          …      …
284            NaN    West Bank and Gaza  31.952200   35.233200        0
285            NaN  Winter Olympics 2022  39.904200  116.407400        0
286            NaN                 Yemen  15.552727   48.516388        0
287            NaN                Zambia -13.133897   27.849332        0
288            NaN              Zimbabwe -19.015438   29.154857        0

     1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  …  2/28/23  3/1/23  \
0          0        0        0        0        0  …   209322  209340
1          0        0        0        0        0  …   334391  334408
2          0        0        0        0        0  …   271441  271448
3          0        0        0        0        0  …    47866   47875
4          0        0        0        0        0  …   105255  105277
..       …      …      …      …      …  …       …       …
284        0        0        0        0        0  …   703228  703228
285        0        0        0        0        0  …      535     535
286        0        0        0        0        0  …    11945   11945
287        0        0        0        0        0  …   343012  343012
288        0        0        0        0        0  …   263921  264127

     3/2/23  3/3/23  3/4/23  3/5/23  3/6/23  3/7/23  3/8/23  3/9/23
0    209358  209362  209369  209390  209406  209436  209451  209451
1    334408  334427  334427  334427  334427  334427  334443  334457
2    271463  271469  271469  271477  271477  271490  271494  271496
3     47875   47875   47875   47875   47875   47875   47890   47890
4    105277  105277  105277  105277  105277  105277  105288  105288
..      …       …       …       …       …       …       …
284  703228  703228  703228  703228  703228  703228  703228  703228
285     535     535     535     535     535     535     535     535
286   11945   11945   11945   11945   11945   11945   11945   11945
287  343079  343079  343079  343135  343135  343135  343135  343135
288  264127  264127  264127  264127  264127  264127  264276  264276

[201 rows x 1147 columns]
```

### 1.21 Concatenate rows

[14]:
```
covid_us = covid_19[covid_19['Country/Region']=='US']
covid_afg = covid_19[covid_19['Country/Region'] == 'Afghanistan']

covid_afg

covid_concat = pd.concat([covid_us,covid_afg])
covid_concat
```

[14]:
```
    Province/State Country/Region       Lat        Long  1/22/20  1/23/20  \
260            NaN             US  40.00000 -100.000000        1        1
0              NaN    Afghanistan  33.93911   67.709953        0        0

     1/24/20  1/25/20  1/26/20  1/27/20  …    2/28/23     3/1/23     3/2/23  \
260        2        2        5        5  …  103443455  103533872  103589757
0          0        0        0        0  …     209322     209340     209358

        3/3/23     3/4/23     3/5/23     3/6/23     3/7/23     3/8/23  \
260  103648690  103650837  103646975  103655539  103690910  103755771
0       209362     209369     209390     209406     209436     209451

        3/9/23
260  103802702
0       209451

[2 rows x 1147 columns]
```

### 1.22 Transforming data using stack to convert it to columns

[15]:
```
covid_us = covid_19[covid_19['Country/Region'] == 'US']
covid_drop = covid_us.drop(['Lat','Long'],axis=1)
covid_drop
covid_transform = covid_drop.stack()
covid_transformF=covid_transform.reset_index()
covid_transformF
```

[15]:
```
      level_0         level_1          0
0         260  Country/Region         US
1         260         1/22/20          1
2         260         1/23/20          1
3         260         1/24/20          2
4         260         1/25/20          2
…         …              …          …
1139      260          3/5/23  103646975
1140      260          3/6/23  103655539
1141      260          3/7/23  103690910
```

```
1142      260          3/8/23  103755771
1143      260          3/9/23  103802702

[1144 rows x 3 columns]
```

### 1.23 Renaming Columns

[16]:
```
covid_transformF = covid_transformF[covid_transformF['level_1']!= 'Country/
↪Region']
covid_transform_filter = covid_transformF[['level_1',0]]
covid_transform_filter.columns = ['Date', 'Number of Cases']
covid_transform_filter
```

[16]:
```
          Date Number of Cases
1     1/22/20               1
2     1/23/20               1
3     1/24/20               2
4     1/25/20               2
5     1/26/20               5
…        …              …
1139   3/5/23       103646975
1140   3/6/23       103655539
1141   3/7/23       103690910
1142   3/8/23       103755771
1143   3/9/23       103802702

[1143 rows x 2 columns]
```

[17]:
```
covid_sort = covid_transform_filter.sort_values(by = 'Number of␣
↪Cases',ascending = False)
covid_sort
```

[17]:
```
          Date Number of Cases
1143   3/9/23       103802702
1142   3/8/23       103755771
1141   3/7/23       103690910
1140   3/6/23       103655539
1138   3/4/23       103650837
…        …              …
6     1/27/20               5
3     1/24/20               2
4     1/25/20               2
2     1/23/20               1
1     1/22/20               1

[1143 rows x 2 columns]
```

### 1.24 Parsing date in pandas

[18]:
```
covid_sort['Date'] = pd.to_datetime(covid_sort['Date'])
covid_sort = covid_sort.set_index('Date')
covid_sort
```

```
/tmp/ipykernel_2327/2480021473.py:1: UserWarning: Could not infer format, so
each element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  covid_sort['Date'] = pd.to_datetime(covid_sort['Date'])
```

[18]:

| Date | Number of Cases |
|---|---|
| 2023-03-09 | 103802702 |
| 2023-03-08 | 103755771 |
| 2023-03-07 | 103690910 |
| 2023-03-06 | 103655539 |
| 2023-03-04 | 103650837 |
| … | … |
| 2020-01-27 | 5 |
| 2020-01-24 | 2 |
| 2020-01-25 | 2 |
| 2020-01-23 | 1 |
| 2020-01-22 | 1 |

[1143 rows x 1 columns]

[19]:
```
covid_group = covid_sort.groupby(pd.Grouper(freq='SME'))['Number of Cases'].
 ↪sum().reset_index().sort_values(by='Number of Cases', ascending=False)
covid_group
```

[19]:

| | Date | Number of Cases |
|---|---|---|
| 72 | 2023-01-15 | 1632531390 |
| 70 | 2022-12-15 | 1604235865 |
| 66 | 2022-10-15 | 1555152315 |
| 73 | 2023-01-31 | 1540521761 |
| 71 | 2022-12-31 | 1518378713 |
| .. | … | … |
| 4 | 2020-03-15 | 918459 |
| 3 | 2020-02-29 | 10942 |
| 2 | 2020-02-15 | 214 |
| 1 | 2020-01-31 | 171 |
| 0 | 2020-01-15 | 33 |

[76 rows x 2 columns]

### 1.25 Taking Substrings and Parsing Data

[20]:
```
covid_group['Date'] = covid_group['Date'].astype(str)
covid_group['New Date'] = covid_group['Date'].str[0:7]
covid_group['New Date'] = covid_group['New Date'].replace({'-':'/'},regex =True)
covid_group
covid_final =covid_group[['Number of Cases','New Date']]
covid_final_head = covid_final.head(1)
covid_final_head
covid_final_top = covid_final.head(10)
covid_final_top10 = covid_final_top.sort_values(by="New Date", ascending=True)
covid_final_top10
```

[20]:

| | Number of Cases | New Date |
|---|---|---|
| 62 | 1500087842 | 2022/08 |
| 65 | 1449244676 | 2022/09 |
| 66 | 1555152315 | 2022/10 |
| 67 | 1467012163 | 2022/10 |
| 69 | 1488536116 | 2022/11 |
| 68 | 1476303690 | 2022/11 |
| 70 | 1604235865 | 2022/12 |
| 71 | 1518378713 | 2022/12 |
| 72 | 1632531390 | 2023/01 |
| 73 | 1540521761 | 2023/01 |

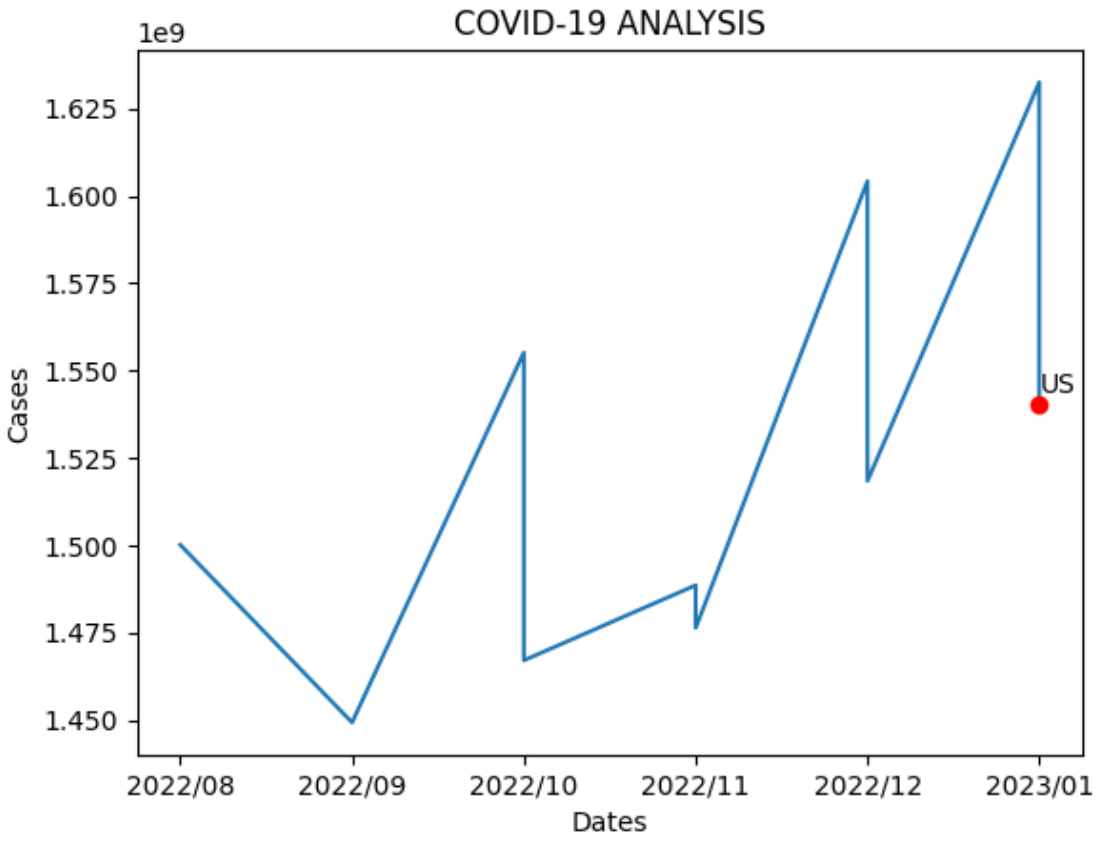
### 1.26 Data Visualization

[21]:
```
import matplotlib
from matplotlib import pyplot as plt
x_axis = covid_final_top10['New Date']
y_axis = covid_final_top10['Number of Cases']
figure_1 = plt.figure()
axis = figure_1.add_subplot(111)
line, =axis.plot(x_axis,y_axis)

x_max = max(x_axis)
y_pos = y_axis.index.max()
y_max=y_axis[y_pos]

print(x_max)
print(y_max)

axis.annotate('US' ,xy = (x_max,y_max + 3000000), xytext = (x_max,y_max +␣
 ↪3000000))
axis.plot(x_max,y_max,'ro')
plt.xlabel('Dates')
plt.ylabel('Cases')
```

```
plt.title("COVID-19 ANALYSIS")
plt.show
```

```
2023/01
1540521761
```

[21]: <function matplotlib.pyplot.show(close=None, block=None)>




### 1.27 Selecting 5 countries that has highest number of cases

[22]:
```
covid_melt = covid_drop_main.melt(id_vars = ['Country/Region'])
covid_melt_1 = covid_melt.dropna()
covid_melt_state = covid_melt_1[covid_melt_1['variable'] != 'Province/State']
covid_melt_state_final = covid_melt_state.groupby(['Country/Region'])['value'].
  ↪sum().reset_index().sort_values(by='value',ascending = False)
covid_melt_state_final
```

[22]:

| | Country/Region | value |
|---|---|---|
| 186 | US | 53813184406 |
| 80 | India | 29131119694 |

```
24                  Brazil  21182690594
63                  France  16105911886
67                 Germany  13686043720
..                     …          …
197  Winter Olympics 2022       214462
76                Holy See        26807
107            MS Zaandam         9665
5              Antarctica         4961
93           Korea, North          300

[201 rows x 2 columns]
```

### 1.28 Selecting 5 countries

[23]:
```
covid_top5_countries = covid_melt_state.loc[covid_melt_state['Country/Region'].
 ↪isin(['US','India','Brazil','France','Germany'])]
set(covid_top5_countries['Country/Region'])
```

[23]:
```
{'Brazil', 'France', 'Germany', 'India', 'US'}
```

### 1.29 Renaming columns grouping and parsing data

[24]:
```
import pandas as pd


pd.set_option('mode.chained_assignment', None)

top_countries = covid_melt_state_final['Country/Region'].head(5).tolist()


filtered_data = covid_melt_state[covid_melt_state['Country/Region'].
 ↪isin(top_countries)].copy()


filtered_data.columns = ['Country', 'Date of Cases', 'Number of Cases']


filtered_data['Date of Cases'] = pd.to_datetime(filtered_data['Date of Cases'])


aggregated_data = (
    filtered_data
    .groupby(['Country', pd.Grouper(key='Date of Cases', freq='SME')])['Number␣
 ↪of Cases']
    .sum()
    .reset_index()
```

```
)


print(set(aggregated_data['Country']))
```

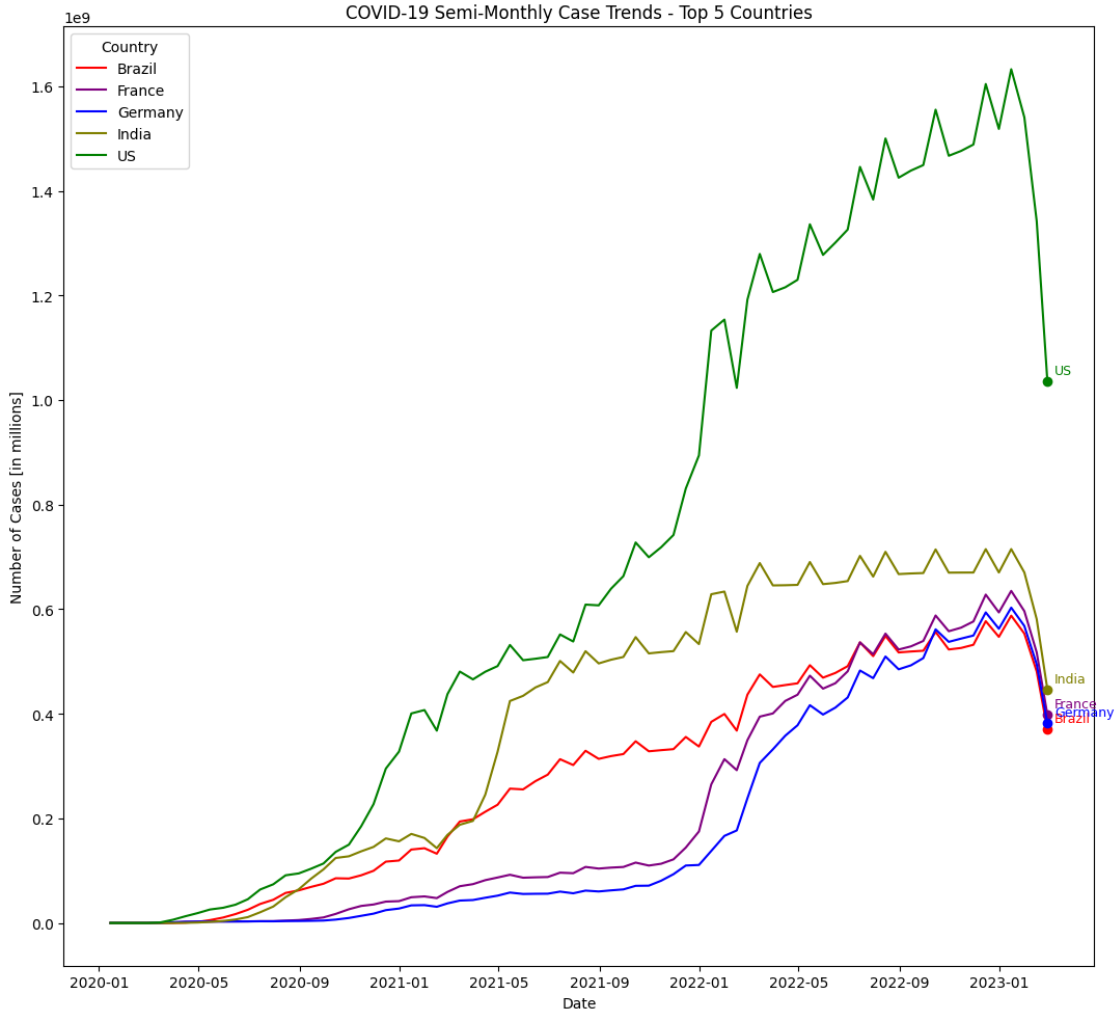
```
{'India', 'France', 'US', 'Brazil', 'Germany'}
```

```
/tmp/ipykernel_2327/3167260892.py:15: UserWarning: Could not infer format, so
each element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  filtered_data['Date of Cases'] = pd.to_datetime(filtered_data['Date of
Cases'])
```

### 1.30 Data Visualization

[25]:
```
import matplotlib.pyplot as plt

country_colors = {
    'Brazil': 'red',
    'US': 'green',
    'India': 'olive',
    'Germany': 'blue',
    'France': 'purple'
}

fig, ax = plt.subplots(figsize=(11, 10))

for country in aggregated_data['Country'].unique():
    country_data = aggregated_data[aggregated_data['Country'] == country]
    ax.plot(
        country_data['Date of Cases'],
        country_data['Number of Cases'],
        label=country,
        color=country_colors.get(country, 'black')
    )


    latest = country_data.iloc[-1]


    ax.plot(
        latest['Date of Cases'],
        latest['Number of Cases'],
        marker='o',
        color=country_colors.get(country, 'black')
    )
```

```
    ax.annotate(
        country,
        xy=(latest['Date of Cases'], latest['Number of Cases']),
        xytext=(5, 5),
        textcoords='offset points',
        fontsize=9,
        color=country_colors.get(country, 'black')
    )

ax.set_title('COVID-19 Semi-Monthly Case Trends - Top 5 Countries')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Cases [in millions]')
ax.legend(title='Country')
plt.tight_layout()
plt.show()
```



[26]:
```
covid_us = aggregated_data[aggregated_data['Country'] == 'US']
covid_us['Number of Cases'].apply(lambda x : "{:,}".format(x))
```

[26]:
```
304               33
305              171
306              214
307           10,942
308          918,459
             ...
375    1,518,378,713
376    1,632,531,390
377    1,540,521,761
378    1,342,063,007
379    1,036,418,508
Name: Number of Cases, Length: 76, dtype: object
```

### 1.31 Ploting graph using Plotly library

[27]:
```
import plotly
```

[50]:
```
!pip install plotly
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting plotly
  Downloading plotly-6.0.1-py3-none-any.whl (14.8 MB)
                             14.8/14.8 MB
4.5 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: packaging in ./.local/lib/python3.10/site-
packages (from plotly) (24.2)
Collecting narwhals>=1.15.1
  Downloading narwhals-1.35.0-py3-none-any.whl (325 kB)
                            325.7/325.7

KB 4.5 MB/s eta 0:00:0000:0100:01
Installing collected packages: narwhals, plotly
Successfully installed narwhals-1.35.0 plotly-6.0.1
```

[28]:
```
import plotly.graph_objects as go
import numpy as np

dates = covid_us['Date of Cases']
cases = covid_us['Number of Cases']


selected_indices = np.round(np.linspace(0, len(cases) - 1, 10)).astype(int)

filtered_dates = dates.iloc[selected_indices]
```

```
filtered_cases = cases.iloc[selected_indices]

fig = go.Figure(
    data=[
        go.Scatter(
            x=filtered_dates,
            y=filtered_cases,
            mode="lines+markers+text",
            name="US Case Data",
            text=filtered_cases,
            textposition="top center",
            line=dict(color='darkmagenta')
        )
    ],
    layout={
        "title": "COVID-19 Case Progression in the United States (Sampled)",
        "xaxis": {"title": "Date"},
        "yaxis": {"title": "Confirmed Cases"}
    }
)

fig.show()
```

### 1.32 Covid 19 daily trend using plotly

[72]:
```
import plotly.graph_objects as go

# Prepare the dataset
covid_filtered = covid_us[['Date of Cases', 'Number of Cases']].copy()
covid_filtered = covid_filtered.sort_values(by='Date of Cases')

x_values = covid_filtered['Date of Cases']
y_values = covid_filtered['Number of Cases']


y_min = y_values.min()
y_max = y_values.max()

animation_frames = []
for i in range(1, len(x_values) + 1):
    frame = go.Frame(
        data=[
            go.Scatter(
                x=x_values[:i],
                y=y_values[:i],
                mode="lines",
                line=dict(color='darkmagenta'),
```

```
                name="COVID-19 Trend"
            )
        ]
    )
    animation_frames.append(frame)


figure = go.Figure(
    data=[
        go.Scatter(
            x=[x_values.iloc[0]],
            y=[y_values.iloc[0]],
            mode="lines",
            line=dict(color='darkmagenta'),
            name="COVID-19 Trend"
        )
    ],
    layout={
        "title": "COVID-19 Spread in the US Over Time",
        "xaxis": {"title": "Date", "range": [x_values.min(), x_values.max()]},
        "yaxis": {"title": "Confirmed Cases", "range": [y_min, y_max]},
        "updatemenus": [
            {
                "type": "buttons",
                "buttons": [
                    {
                        "label": "Play",
                        "method": "animate",
                        "args": [None, {"frame": {"duration": 100, "redraw":␣
 ↪False}, "fromcurrent": True}]
                    }
                ]
            }
        ]
    },
    frames=animation_frames
)

figure.show()
```

[ ]:

[70]:
```
covid_sort = covid_us[['Date of Cases', 'Number of Cases']].copy()
print(covid_sort.columns)
```

```
Index(['Date of Cases', 'Number of Cases'], dtype='object')
```

[5]:
```
# Global Pandemic Visualization Dashboard
# Author: Krish
# Date: 11/04/25

import pandas as pd
import plotly.express as px
from dash import Dash, dcc, html, Input, Output

def load_pandemic_data():
    """Load and process global pandemic dataset"""
    data_url = (
        'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/'
        'master/csse_covid_19_data/csse_covid_19_time_series/'
        'time_series_covid19_confirmed_global.csv'
    )


    raw_data = pd.read_csv(data_url)
    processed_data = raw_data.drop(
        ['Province/State', 'Lat', 'Long'],
        axis=1
    )


    reshaped_data = pd.melt(
        processed_data,
        id_vars=['Country/Region'],
        var_name='Day',
        value_name='ConfirmedCases'
    )


    reshaped_data.columns = ['Territory', 'DateRecorded', 'CaseVolume']
    reshaped_data['DateRecorded'] = pd.
 ↪to_datetime(reshaped_data['DateRecorded'])

    return reshaped_data

def aggregate_data(df, frequency='SME'):
    """Aggregate data by specified time frequency"""
    aggregated = df.groupby(
        [pd.Grouper(key='DateRecorded', freq=frequency), 'Territory']
    )['CaseVolume'].sum().reset_index()


    recent_data = aggregated[aggregated['DateRecorded'] <= '2023-01-15']
    return recent_data.sort_values('DateRecorded')
```

```
def create_dashboard_layout(dataframe):
    """Generate the dashboard interface"""
    return html.Div(
        className='dashboard-container',
        children=[
            html.Div(
                className='visualization-panel',
                children=[
                    dcc.Graph(id='pandemic_trend_visualization')
                ]
            ),
            html.Div(
                className='control-panel',
                children=[
                    html.Br(),
                    html.Label(
                        'Compare National Trends:',
                        className='control-label'
                    ),
                    dcc.Dropdown(
                        id='nation_selector',
                        options=[
                            {'label': loc, 'value': loc}
                            for loc in dataframe['Territory'].unique()
                        ],
                        value=['United States', 'Germany', 'Japan'],
                        multi=True,
                        searchable=True,
                        clearable=False,
                        placeholder='Select nations...',
                        className='nation-dropdown'
                    )
                ]
            )
        ]
    )

def configure_callbacks(app, dataframe):
    """Set up dashboard interactivity"""
    @app.callback(
        Output('pandemic_trend_visualization', 'figure'),
        Input('nation_selector', 'value')
    )
    def update_visualization(selected_nations):
        if not selected_nations:
            return px.line(title='Please select nations to visualize')
```

```
        filtered_data = dataframe[dataframe['Territory'].isin(selected_nations)]

        visualization = px.line(
            filtered_data,
            x='DateRecorded',
            y='CaseVolume',
            color='Territory',
            title='Global Pandemic Trends (Semi-Monthly Aggregation)',
            labels={
                'DateRecorded': 'Observation Period',
                'CaseVolume': 'Confirmed Cases',
                'Territory': 'Geographic Region'
            }
        )

        visualization.update_layout(
            template='plotly_dark',
            hovermode='x unified',
            xaxis_title='Timeline',
            yaxis_title='Case Count',
            legend_title='Selected Nations'
        )

        return visualization

def main():
    """Main application execution"""

    initial_data = load_pandemic_data()
    analysis_data = aggregate_data(initial_data)


    dashboard_app = Dash(__name__)
    dashboard_app.layout = create_dashboard_layout(analysis_data)
    configure_callbacks(dashboard_app, analysis_data)


    dashboard_app.run(debug=True)

if __name__ == '__main__':
    main()
```

```
/tmp/ipykernel_5457/2143295073.py:34: UserWarning:

Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
```

```
specify a format.


<IPython.lib.display.IFrame at 0x7f78910f2ef0>
```

[6]:
```
import pandas as pd
import plotly.express as px
from dash import Dash, dcc, html, Input, Output


def get_processed_covid_data():

    data_url = ('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/'
                'master/csse_covid_19_data/csse_covid_19_time_series/'
                'time_series_covid19_confirmed_global.csv')


    raw_df = pd.read_csv(data_url)
    cleaned_df = raw_df.drop(columns=['Province/State', 'Lat', 'Long'])


    melted_df = cleaned_df.melt(id_vars='Country/Region')
    melted_df.columns = ['Location', 'ObservationDate', 'TotalCases']
    melted_df['ObservationDate'] = pd.to_datetime(melted_df['ObservationDate'])


    monthly_aggregated = melted_df.groupby(
        [pd.Grouper(key='ObservationDate', freq='ME'), 'Location']
    )['TotalCases'].sum().reset_index()


    final_data = monthly_aggregated[monthly_aggregated['ObservationDate'] <=␣
 ↪'2023-01-31']
    return final_data.sort_values(by='ObservationDate')


def create_dashboard_app():
    app = Dash(__name__)


    visualization_data = get_processed_covid_data()


    app.layout = html.Div(
        style={'backgroundColor': '#121212'},
        children=[
            html.H1(
                "Global Pandemic Trends Dashboard",
```

```
                style={"textAlign": "center", "color": "#FFFFFF"}
            ),
            html.Div([
                dcc.Tabs(
                    id="navigation-tabs",
                    children=[
                        dcc.Tab(
                            label='Infection Trends',
                            children=[
                                html.Div([
                                    html.H2(
                                        "Monthly Infection Statistics",
                                        style={
                                            'textAlign': 'center',
                                            'color': '#FFFFFF'
                                        }
                                    ),
                                    html.Label(
                                        "Select Locations:",
                                        style={
                                            'color': '#FFFFFF',
                                            'fontSize': '18px'
                                        }
                                    ),
                                    dcc.Dropdown(
                                        id='location-selector',
                                        options=[
                                            {'label': loc, 'value': loc}
                                            for loc in␣
↪visualization_data['Location'].unique()
                                        ],
                                        value=['United States', 'India',␣
↪'Brazil'],
                                        multi=True,
                                        style={"width": "100%"}
                                    ),
                                    dcc.Graph(id='infection-trend-chart')
                                ], style={'padding': '20px'})
                            ]
                        ),
                        dcc.Tab(
                            label='Temporal Analysis',
                            children=[
                                html.H2(
                                    "Temporal Infection Patterns",
                                    style={
                                        "textAlign": "center",
```

```
                                    "color": "#FFFFFF"
                                }
                            )
                        ]
                    )
                ]
            )
        ])
    ]
)


@app.callback(
    Output('infection-trend-chart', 'figure'),
    Input('location-selector', 'value')
)
def update_visualization(selected_locations):
    filtered_df = visualization_data[
        visualization_data['Location'].isin(selected_locations)
    ]

    fig = px.line(
        filtered_df,
        x='ObservationDate',
        y='TotalCases',
        color='Location',
        height=600,
        labels={
            'ObservationDate': 'Time Period',
            'TotalCases': 'Confirmed Infections',
            'Location': 'Geographical Area'
        }
    )

    fig.update_layout(
        title=dict(
            text='Monthly Infection Trends',
            font=dict(size=24),
            x=0.5
        ),
        hovermode='x unified',
        updatemenus=[dict(
            type='buttons',
            direction='left',
            buttons=[
                dict(
                    args=[{'yaxis.type': 'linear'}],
```

```
                    label='Linear Scale',
                    method='relayout'
                ),
                dict(
                    args=[{'yaxis.type': 'log'}],
                    label='Logarithmic Scale',
                    method='relayout'
                )
            ]
        )],
        xaxis=dict(
            rangeselector=dict(
                buttons=[
                    dict(
                        count=1,
                        label='1 Month',
                        step='month',
                        stepmode='backward'
                    ),
                    dict(
                        count=6,
                        label='6 Months',
                        step='month',
                        stepmode='backward'
                    ),
                    dict(
                        count=1,
                        label='Year to Date',
                        step='year',
                        stepmode='todate'
                    ),
                    dict(
                        count=1,
                        label='1 Year',
                        step='year',
                        stepmode='backward'
                    ),
                    dict(step='all')
                ]
            ),
            rangeslider=dict(visible=True),
            type='date'
        ),
        paper_bgcolor='#121212',
        plot_bgcolor='#121212',
        font_color='#CCCCCC',
        legend_title_text='Geographical Area'
```

```
        )

        return fig

    return app


if __name__ == '__main__':
    dashboard = create_dashboard_app()
    dashboard.run(debug=True, port=8051)
```

```
/tmp/ipykernel_5457/743064335.py:19: UserWarning:

Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.


<IPython.lib.display.IFrame at 0x7f7890fe30a0>
```

[ ]: