Chattapadhyay_DSC540_Milestone2

Date: Jan 15, 2023

Project DSC540 - Covid 19 Data Analysis and Comparison

Author: Kausik Chattapadhyay

Milestone 2

Perform at least 5 data transformation and/or cleansing steps to your flat file data. For example:

• Replace Headers

• Format data into a more readable format

• Identify outliers and bad data

• Find duplicates

• Fix casing or inconsistent values

• Conduct Fuzzy Matching

Dataset

CSV – The Covid 19 data is scrapped from John Hopkins University github repo : https://github.com/CSSEGISandData/COVID19/tree/master/csse_covid_19_data/csse_covid_19_time_series , this has

- Daily time series summary tables, including confirmed, deaths and recovered. All data is read in from the daily case report. The time series tables are subject to be updated if inaccuracies are identified in our historical data.
- Two time series tables are for the US confirmed cases and deaths, reported at the county level.
- Three time series tables are for the global confirmed cases, recovered cases and deaths. Australia, Canada and China are reported at the province/state level.
- Data is updated at a daily basis.

Load the necessary libraries.

In [2]:
```
# import libraries
from datetime import datetime, timedelta   # for date and time opeations
import os   # for file and folder operations
import re   # for regular expression operations
import glob # for listing files in a folder
import requests  # for getting web contents
import pandas as pd  # storing and analysing data
from bs4 import BeautifulSoup # for scraping web contents
import numpy as np # numerical analysis
```

In [3]:
```
# Read dataset
conf_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
deaths_df = pd.read_csv('time_series_covid19_deaths_global.csv')
recv_df = pd.read_csv('time_series_covid19_recovered_global.csv')
```

In [4]:
```
#View the data for any cleanup
conf_df.head()
```

Out[4]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	1/6/23	1/7/23	1/8/23	1/9/23	1/10/23	1/11/23	1/12/23	1/13/23	1/14/23	1/15/23
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	207766	207766	207819	207841	207866	207900	207900	207900	207900	207900
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	333887	333916	333947	333948	333995	333995	334018	334018	334029	334037
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	271250	271254	271254	271255	271262	271268	271277	271286	271287	271287
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	47781	47781	47781	47781	47781	47781	47781	47781	47781	47781
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	105095	105095	105095	105095	105095	105095	105095	105095	105095	105095

5 rows × 1094 columns

In [5]:
```
deaths_df.head()
```

Out[5]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	1/6/23	1/7/23	1/8/23	1/9/23	1/10/23	1/11/23	1/12/23	1/13/23	1/14/23	1/15/23
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	7850	7850	7853	7854	7854	7854	7854	7854	7854	7854
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	3596	3596	3596	3596	3596	3596	3596	3596	3596	3596
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	6881	6881	6881	6881	6881	6881	6881	6881	6881	6881
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	165	165	165	165	165	165	165	165	165	165
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	1930	1930	1930	1930	1930	1930	1930	1930	1930	1930

5 rows × 1094 columns

In [6]:
```
recv_df.head()
```

Out[6]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	1/6/23	1/7/23	1/8/23	1/9/23	1/10/23	1/11/23	1/12/23	1/13/23	1/14/23	1/15/23
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 1094 columns

In [7]:
```
# Merge the datasets
# extract dates
dates = conf_df.columns[4:]

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

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

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

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

```
(315910, 6)
(315910, 6)
(298660, 6)
```

In [8]:
```
# merge dataframes to get a full dataframe, we will then perform a cleanup on , the final dataset

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[8]:

	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

Cleanup Activities

In [9]:
```
# 1. Convert to proper date format
full_table['Date'] = pd.to_datetime(full_table['Date'])

# 2. fill na with 0
full_table['Recovered'] = full_table['Recovered'].fillna(0)

# 3. convert to int datatype

full_table['Recovered'] = full_table['Recovered'].astype('int')

# 4. fixing Country names
# 4.1 renaming countries, regions, provinces
full_table['Country/Region'] = full_table['Country/Region'].replace('Korea, South', 'South Korea')

# 4.2 Greenland
full_table.loc[full_table['Province/State']=='Greenland', 'Country/Region'] = 'Greenland'

# 4.3 Mainland china to China

full_table['Country/Region'] = full_table['Country/Region'].replace('Mainland China', 'China')

# 5. Removing county wise data to avoid double counting

full_table = full_table[full_table['Province/State'].str.contains(',')!=True]
```

Adding Calculated values and filling missing values

In [10]:
```
# Active Case = confirmed - deaths - recovered
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']

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

# fill missing numerical values with 0
cols = ['Confirmed', 'Deaths', 'Recovered', 'Active']
full_table[cols] = full_table[cols].fillna(0)

# fixing datatypes
full_table['Recovered'] = full_table['Recovered'].astype(int)

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

Out[10]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
22948		Ethiopia	9.145000	40.489700	2020-04-10	65	3	4	58
277741	Queensland	Australia	-27.469800	153.025100	2022-09-09	1620034	2048	0	1617986
220762		Tonga	-21.179000	-175.198200	2022-02-23	289	0	0	289
190269		Djibouti	11.825100	42.590300	2021-11-10	13493	186	0	13307
217464		Ghana	7.946500	-1.023200	2022-02-12	157751	1419	0	156332
235526		Uzbekistan	41.377491	64.585262	2022-04-15	238214	1637	0	236577

In [11]:
```
# function to change value of a column in dataframe

def change_val(date, ref_col, val_col, dtnry):
    for key, val in dtnry.items():
        full_table.loc[(full_table['Date']==date) & (full_table[ref_col]==key), val_col] = val
```

In [12]:
```
# checking values
full_table[(full_table['Date']=='2/12/20') & (full_table['Province/State']=='Hubei')]
```

Out[12]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
6141	Hubei	China	30.9756	112.2707	2020-02-12	33366	1068	2686	29612

Removing Outliers

In [13]:
```
# there is ship rows info which contains ships with Covid-19 reported cases
# this is an outlier for our analysis so we will remove that info from our dataframe
# ship rows containing ships with COVID-19 reported cases
ship_rows = (full_table['Province/State'].str.contains('Grand Princess') | full_table['Province/State'].str.contains('Diamond Princess') |  full_table['Country/Region']

# ship
ship = full_table[ship_rows]

# Latest cases from the ships
ship_latest = ship[ship['Date']==max(ship['Date'])]

# ship_latest.style.background_gradient(cmap='Pastel1_r')
# skipping rows with ships info
full_table = full_table[~(ship_rows)]
```

In [14]:
```
full_table
```

Out[14]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
0		Afghanistan	33.939110	67.709953	2020-01-22	0	0	0	0
1		Albania	41.153300	20.168300	2020-01-22	0	0	0	0
2		Algeria	28.033900	1.659600	2020-01-22	0	0	0	0
3		Andorra	42.506300	1.521800	2020-01-22	0	0	0	0
4		Angola	-11.202700	17.873900	2020-01-22	0	0	0	0
...	...	...	...	...	...	...	...	...	...
315005		West Bank and Gaza	31.952200	35.233200	2023-01-15	703228	5708	0	697520
315006		Winter Olympics 2022	39.904200	116.407400	2023-01-15	535	0	0	535
315007		Yemen	15.552727	48.516388	2023-01-15	11945	2159	0	9786
315008		Zambia	-13.133897	27.849332	2023-01-15	337303	4036	0	333267
315009		Zimbabwe	-19.015438	29.154857	2023-01-15	259981	5637	0	254344

308470 rows × 9 columns

In [15]:
```
full_table.isna().sum()
```

Out[15]:

```
Province/State      0
Country/Region      0
Lat              2180
Long             2180
Date                0
Confirmed           0
Deaths              0
Recovered           0
Active              0
dtype: int64
```

In [16]:
```
full_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 308470 entries, 0 to 315009
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Province/State  308470 non-null  object
 1   Country/Region  308470 non-null  object
 2   Lat             306290 non-null  float64
 3   Long            306290 non-null  float64
 4   Date            308470 non-null  datetime64[ns]
 5   Confirmed       308470 non-null  int64
 6   Deaths          308470 non-null  int64
 7   Recovered       308470 non-null  int64
 8   Active          308470 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(4), object(2)
memory usage: 23.5+ MB
```

In [18]:
```
full_table['Country/Region'].value_counts()
```

Out[18]:

```
China                37060
Canada               15260
United Kingdom       15260
France               13080
Australia             8720
                     ...
Guinea                1090
Guinea-Bissau         1090
Guyana                1090
Haiti                 1090
Zimbabwe              1090
Name: Country/Region, Length: 200, dtype: int64
```

In [ ]:
```

```