COVID-19
CASES ANALYSIS

INTRODUCTION:

The project involves analysing COVID-19 cases and deaths data using IBM Cognos with the main goal of comparing mean values and standard deviations of cases and deaths per day and by country in the EU/EEA (European Union/European Economic Area).

SOURCE CODE:

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import plotly.express as px
```

In [2]:

```
eur_df_new                                                         =

pd.read_csv('../input/covid19-cases-in-africa/covid19_europe.csv')

#import europe Cases
```

In [3]:

```
eur_df_new.head()
```

Out[3]:

	ObservationDate	Country_Region	Province_State	Confirmed	Deaths	Recovered	Active
0	2020-01-24	France	NaN	2	0	0	0.0
1	2020-01-25	France	NaN	3	0	0	0.0
2	2020-01-26	France	NaN	3	0	0	0.0
3	2020-01-27	France	NaN	3	0	0	0.0
4	2020-01-28	France	NaN	4	0	0	0.0

In [4]:

```
eur_df_new.shape
#32138 records, 7 columns. Now we can dive more into the columns and their contents.
```

Out[4]:

```
(41553, 7)
```

In [5]:

```
eur_df_new.dtypes

#data types
```

Out[5]:

```
ObservationDate      object
Country_Region       object
Province_State       object
Confirmed             int64
Deaths                int64
Recovered             int64
Active              float64
dtype: object
```

In [6]:

```
pd.unique(eur_df_new['Country_Region'])

#this dataset looks at europe
```

Out[6]:

```
array(['France', 'Germany', 'Finland', 'Italy', 'United Kingdom',
       'Russia', 'Sweden', 'Spain', 'Belgium', 'Austria', 'Croatia',
       'Switzerland', 'Greece', 'North Macedonia', 'Norway', 'Romania',
       'Denmark', 'Estonia', 'Netherlands', 'San Marino', 'Belarus',
       'Iceland', 'Lithuania', 'Ireland', 'Luxembourg', 'Monaco',
       'Czechia', 'Portugal', 'Andorra', 'Latvia', 'Ukraine', 'Hungary',
       'Liechtenstein', 'Poland', 'Bosnia and Herzegovina', 'Slovenia',
       'Serbia', 'Slovakia', 'Vatican City', 'Malta', 'Bulgaria',
       'Moldova', 'Albania', 'Holy See', 'Guernsey', 'Jersey',
       'Montenegro'], dtype=object)
```

In [7]:

```
eur_df_new.isnull().sum()
#Missing Value Count.
#7515 states or provinces within a country missing here., 24 active cases missing.
```

Out[7]:

```
ObservationDate        0
Country_Region         0
Province_State      8955
Confirmed              0
Deaths                 0
Recovered              0
Active                24
dtype: int64
```

In [8]:

```
eur_df_new['Province_State'].isnull().sum()/25682
#This depicts the percentage of the Province_States values that are missing.
#The threshold I go by is that if upwards of 25-30% of the values are missing I drop the column.
```

Out[8]:

```
0.3486877969005529
```

In [9]:

```python
eur_df_new = eur_df_new.drop(columns = 'Province_State')
```

In [10]:

```python
eur_df_new.isnull().values.any()
```

Out[10]:

```
True
```

In [11]:

```python
eur_df_new = eur_df_new.dropna()
```

In [12]:

```python
eur_df_new.shape #new shape
```

Out[12]:

```
(41529, 6)
```

In [13]:

```
import datetime as dt
#use it to obtain month and year in column for potential grouping purposes
```

In [14]:

```
eur_df_new['ObservationDate'] = pd.to_datetime(eur_df_new['ObservationDate'])
eur_df_new['mnth_yr'] = eur_df_new['ObservationDate'].apply(lambda x: x.strftime('%m-%Y'))

#change datetime format
```

In [15]:

```
eur_df_new.dtypes
#new data types, the datetime conversion was successful
```

Out[15]:

```
ObservationDate    datetime64[ns]
Country_Region             object
Confirmed                   int64
Deaths                      int64
Recovered                   int64
Active                    float64
mnth_yr                    object
dtype: object
```

In [16]:

```
eur_df_new.head()
```

Out[16]:

	ObservationDate	Country_Region	Confirmed	Deaths	Recovered	Active	mnth_yr
0	2020-01-24	France	2	0	0	0.0	01-2020
1	2020-01-25	France	3	0	0	0.0	01-2020
2	2020-01-26	France	3	0	0	0.0	01-2020
3	2020-01-27	France	3	0	0	0.0	01-2020
4	2020-01-28	France	4	0	0	0.0	01-2020

In [17]:

```
eur_df_new = eur_df_new.sort_values(by = 'mnth_yr', ascending=True)
```

In [18]:

```
eur_df_new
```

```
#new column entry successful
```

```
Out[18]:
```

	ObservationDate	Country_Region	Confirmed	Deaths	Recovered	Active	mnth_yr
0	2020-01-24	France	2	0	0	0.0	01-2020
26	2020-01-31	Russia	2	0	0	2.0	01-2020
27	2020-01-31	Finland	1	0	0	1.0	01-2020
28	2020-01-31	Italy	2	0	0	2.0	01-2020
29	2020-01-31	Russia	2	0	0	2.0	01-2020
...	...	...	...	...	...	...	...

39582	2020-10-04	San Marino	732	42	680	10.0	10-2020
39583	2020-10-04	Serbia	33901	754	0	33147.0	10-2020
39584	2020-10-04	Slovakia	13139	55	4828	8256.0	10-2020
39575	2020-10-04	Russia	7483	226	5975	1282.0	10-2020
41552	2020-10-11	United Kingdom	30121	1669	0	28452.0	10-2020

41529 rows × 7 columns

In [19]:

```
eur_df_new[['new_confirmed','new_active','new_deaths','new_recoveries']]
= (eur_df_new.sort_values

(by=['ObservationDate'], ascending=True)

.groupby(['Country_Region'])[['Confirmed','Active','Recovered','Deaths']]
.shift(1))
```

```
#eur_df_new['new_actives'] = eur_df_new['Active'] - eur_df_orig['Active']
#eur_df_new['new_recoveries'] = eur_df_new['Recovered'] - eur_df_orig['Recovered']
#eur_df_new['new_deaths'] = eur_df_new['Deaths']- eur_df_orig['Deaths']
```

In [20]:

```
eur_df_new.head(20)
```

Out[20]:

	ObservationDate	Country_Region	Confirmed	Deaths	Recovered	Active	mnth_yr	new_confirmed	new_active	new_deaths	new_recoveries
0	2020-01-24	France	2	0	0	0.0	01-2020	NaN	NaN	NaN	NaN
26	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
27	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0

28	2020-01-31	Italy	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
29	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
30	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0
31	2020-01-31	Italy	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
32	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
33	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0
34	2020-01-31	Italy	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
25	2020-01-31	Italy	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0

35	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
37	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
38	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0
39	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
40	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0
42	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0
43	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0
44	2020-01-31	Finland	1	0	0	1.0	01-2020	1.0	1.0	0.0	0.0

45	2020-01-31	Russia	2	0	0	2.0	01-2020	2.0	2.0	0.0	0.0

In [21]:

```
eur_df_new.dtypes

#new types
```

Out[21]:

```
ObservationDate    datetime64[ns]
Country_Region             object
Confirmed                   int64
Deaths                      int64
Recovered                   int64
Active                    float64
mnth_yr                    object
new_confirmed             float64
new_active                float64
new_deaths                float64
new_recoveries            float64
dtype: object
```