**Creating Dashboard using Python and Dash Covid-19 Analysis In this webpage we can see daily analysis of covid-19 in graph form**

**Installing necessary library for implementation for reading files,drawing graphs,dash for creating webpage**

In [2]:

```
#!pip install pandas
#!pip install plotly
#!pip install dash
#!pip install dash_bootstrap_components
```

In [3]:

```
import pandas as pd
pd.set_option('max_rows',20)
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "browser"
```

**importing libaries from dash for creating a interactive webpage with different styles,font,colour**

In [4]:

```
import dash
from dash.dependencies import Input, Output
from dash import  dcc
from dash import  html
import dash_bootstrap_components as dbc
```

**here we are reading 3 dataset one is number of confirmed cases second is number of death in a day and third is number of people recoverd**

In [5]:

```
CONF_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
DEAD_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv'
RECV_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv'
```

**reading all three dataset using pandas library**

In [6]:

```
covid_conf_ts = pd.read_csv(CONF_URL)
covid_dead_ts = pd.read_csv(DEAD_URL)
covid_recv_ts = pd.read_csv(RECV_URL)
```

**now we are working on Data Processing part the first function is converting the raw data of country which you select from the dropdown option into a time data series of 1 day. we will truncate first 4 values as it contains general information of country not the data**

In [7]:

```
def process_data(data,cntry='US',window=3):
    conf_ts = data
    conf_ts_cntry = conf_ts[conf_ts['Country/Region']==cntry]
    final_dataset = conf_ts_cntry.T[4:].sum(axis='columns').diff().rolling(window=window
).mean()[40:]
    df = pd.DataFrame(final_dataset,columns=['Total'])
    return df
```

**now in this we will get the total of all three cases. and will display the data for all the country world wide**

In [8]:

```
def get_overall_total(df):
    return df.iloc[:,300].sum()

conf_overall_total = get_overall_total(covid_conf_ts)
dead_overall_total = get_overall_total(covid_dead_ts)
recv_overall_total = get_overall_total(covid_recv_ts)
print('Overall Confirmed:',conf_overall_total)
print('Overall Dead:',dead_overall_total)
print('Overall Recovered:',recv_overall_total)
```

```
Overall Confirmed: 53582011
Overall Dead: 1358385
Overall Recovered: 34481578
```

**now this function is mainly for specific country if you choose from dropdown. it will show all three cases for a particular country.**

In [11]:

```
def get_cntry_total(df,cntry='US'):
    return df[df['Country/Region']==cntry].iloc[:,300].sum()

cntry = 'US'
conf_cntry_total = get_cntry_total(covid_conf_ts,cntry)
dead_cntry_total = get_cntry_total(covid_dead_ts,cntry)
recv_cntry_total = get_cntry_total(covid_recv_ts,cntry)
print(f'{cntry} Confirmed:',conf_cntry_total)
print(f'{cntry} Dead:',dead_cntry_total)
print(f'{cntry} Recovered:',recv_cntry_total)
```

```
US Confirmed: 10847160
US Dead: 245373
US Recovered: 4095146
```

**This is a graph creation part of all three data. will generate line graph using plotly library.**

In [12]:

```
def fig_world_trend(cntry='US',window=3):
    df = process_data(data=covid_conf_ts,cntry=cntry,window=window)
    df.head(10)
    if window==1:
        yaxis_title = "Daily Cases"
    else:
        yaxis_title = "Daily Cases ({}-day MA)".format(window)
    fig = px.line(df, y='Total', x=df.index, title='Daily confirmed cases trend for {}'.
format(cntry),height=600,color_discrete_sequence =['maroon'])
    fig.update_layout(title_x=0.5,plot_bgcolor='#F2DFCE',paper_bgcolor='#F2DFCE',xaxis_ti
tle="Date",yaxis_title=yaxis_title)
    return fig
```

**Dash App we will be using inbuilt function of dash for ceating interactive webpage.**

In [13]:

```
external_stylesheets = [dbc.themes.BOOTSTRAP]
```

**giving the title for webpage**

In [14]:

```
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.title = 'Covid-19 Dashboard by Kartikeya'
```

**Building of page header , giving colour to the header of page, defining font size , background colour etc.**

In [15]:

```
colors = {
    'background': '#111111',
    'bodyColor':'#F2DFCE',
    'text': '#7FDBFF'
}
def get_page_heading_style():
    return {'backgroundColor': colors['background']}


def get_page_heading_title():
    return html.H1(children='COVID-19 Dashboard',
                                        style={
                                        'textAlign': 'center',
                                        'color': colors['text']
                                    })

def get_page_heading_subtitle():
    return html.Div(children='Visualize Covid-19 data generated from sources all over the
world.',
                                         style={
                                             'textAlign':'center',
                                             'color':colors['text']
                                         })

def generate_page_header():
    main_header =  dbc.Row(
                               [
                                   dbc.Col(get_page_heading_title(),md=12)
                               ],
                               align="center",
                               style=get_page_heading_style()
                           )
    subtitle_header = dbc.Row(
                               [
                                   dbc.Col(get_page_heading_subtitle(),md=12)
                               ],
                               align="center",
                               style=get_page_heading_style()
                           )
    header = (main_header,subtitle_header)
    return header
```

**Creating Country Dropdown option, defining inbuilt option of dash to create dropdown menu option, sorting country in alphabetical order.**

In [16]:

```
def get_country_list():
    return covid_conf_ts['Country/Region'].unique()

def create_dropdown_list(cntry_list):
    dropdown_list = []
    for cntry in sorted(cntry_list):
        tmp_dict = {'label':cntry,'value':cntry}
        dropdown_list.append(tmp_dict)
    return dropdown_list

def get_country_dropdown(id):
    return html.Div([
                         html.Label('Select Country'),
                         dcc.Dropdown(id='my-id'+str(id),
                             options=create_dropdown_list(get_country_list()),
                             value='US'
                         ),
                         html.Div(id='my-div'+str(id))
```

```
                    ])
```

**providing graph of particular country which we have made above using plotly library.**

In [17]:

```
def graph1():
    return dcc.Graph(id='graph1',figure=fig_world_trend('US'))
```

**generating cards for overall numbers of all three data. setting all the values of card i.e its font size,alingment,title,name etc .**

In [18]:

```
def generate_card_content(card_header,card_value,overall_value):
    card_head_style = {'textAlign':'center','fontSize':'150%'}
    card_body_style = {'textAlign':'center','fontSize':'200%'}
    card_header = dbc.CardHeader(card_header,style=card_head_style)
    card_body = dbc.CardBody(
        [
            html.H5(f"{int(card_value):,}", className="card-title",style=card_body_style
),
            html.P(
                "Worlwide: {:,}".format(overall_value),
                className="card-text",style={'textAlign':'center'}
            ),
        ]
    )
    card = [card_header,card_body]
    return card
```

**giving 3 values i.e creating 3 cards.**

In [19]:

```
def generate_cards(cntry='US'):
    conf_cntry_total = get_cntry_total(covid_conf_ts,cntry)
    dead_cntry_total = get_cntry_total(covid_dead_ts,cntry)
    recv_cntry_total = get_cntry_total(covid_recv_ts,cntry)
    cards = html.Div(
        [
            dbc.Row(
                [
                    dbc.Col(dbc.Card(generate_card_content("Recovered",recv_cntry_total,
recv_overall_total), color="success", inverse=True),md=dict(size=2,offset=3)),
                    dbc.Col(dbc.Card(generate_card_content("Confirmed",conf_cntry_total,
conf_overall_total), color="warning", inverse=True),md=dict(size=2)),
                    dbc.Col(dbc.Card(generate_card_content("Dead",dead_cntry_total,dead_
overall_total),color="dark", inverse=True),md=dict(size=2)),
                ],
                className="mb-4",
            ),
        ],id='card1'
    )
    return cards
```

**Dash slider for moving average window. we will provide the option according to our feasibility and can slide over it and will get the values according to it.**

In [20]:

```
def get_slider():
    return html.Div([
                        dcc.Slider(
                            id='my-slider',
                            min=1,
                            max=15,
                            step=None,
```

```
                                marks={
                                    1: '1',
                                    3: '3',
                                    5: '5',
                                    7: '1-Week',
                                    14: 'Fortnight'
                                },
                                value=3,
                            ),
                            html.Div([html.Label('Select Moving Average Window')],id='my-div
'+str(id),style={'textAlign':'center'})
                        ])
```

**this is the one of the last part where we generate the APP layout. we will write the sequence in which we want all the things to be dispalyed. firstly we call page header then cards after that we call for dropdown option , graph and then slider.**

In [21]:

```
def generate_layout():
    page_header = generate_page_header()
    layout = dbc.Container(
        [
            page_header[0],
            page_header[1],
            html.Hr(),
            generate_cards(),
            html.Hr(),
            dbc.Row(
                [
                    dbc.Col(get_country_dropdown(id=1),md=dict(size=4,offset=4))

                ]

            ),
            dbc.Row(
                [

                    dbc.Col(graph1(),md=dict(size=6,offset=3))

                ],
                align="center",

            ),
            dbc.Row(
                [
                    dbc.Col(get_slider(),md=dict(size=4,offset=4))
                ]

            ),
        ],fluid=True,style={'backgroundColor': colors['bodyColor']}
    )
    return layout
```

In [22]:

```
app.layout = generate_layout()
```

In [23]:

```
@app.callback(
    [Output(component_id='graph1',component_property='figure'), #line chart
    Output(component_id='card1',component_property='children')], #overall card numbers
    [Input(component_id='my-id1',component_property='value'), #dropdown
     Input(component_id='my-slider',component_property='value')] #slider
)
def update_output_div(input_value1,input_value2):
    return fig_world_trend(input_value1,input_value2),generate_cards(input_value1)
```

In [24]:

```
app.run_server(host= '0.0.0.0',debug=False)
```

```
Dash is running on http://0.0.0.0:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.43.173:8050/ (Press CTRL+C to quit)
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET / HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/deps/prop-types@15.v2_0_0m1638508119.7.2.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/deps/react@16.v2_0_0m1638508119.14.0.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/deps/react-dom@16.v2_0_0m1638508119.14.0.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/dash-renderer/build/dash_renderer.v2_0_0m1638508119.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/deps/polyfill@7.v2_0_0m1638508119.12.1.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/dcc/dash_core_components-shared.v2_0_0m1638508119.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/dash_table/bundle.v5_0_0m1638508119.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/dcc/dash_core_components.v2_0_0m1638508119.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash_bootstrap_components/_components/dash_bootstrap_components.v1_0_1m1638508375.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:21] "GET /_dash-component-suites/dash/html/dash_html_components.v2_0_0m1638508119.min.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-dependencies HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-layout HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_favicon.ico?v=2.0.0 HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-component-suites/dash/dcc/async-graph.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "POST /_dash-update-component HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-component-suites/dash/dcc/async-slider.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-component-suites/dash/dcc/async-dropdown.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:22] "GET /_dash-component-suites/dash/dcc/async-plotlyjs.js HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:27] "POST /_dash-update-component HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:32] "POST /_dash-update-component HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:17:46] "POST /_dash-update-component HTTP/1.1" 200 -
192.168.43.173 - - [04/Dec/2021 18:18:08] "POST /_dash-update-component HTTP/1.1" 200 -
```