COVID-19 Dashboard

Overview

This dashboard provides visualizations of COVID-19 data, allowing users to select a country and a date range to view total cases and deaths. The data is sourced from Our World in Data and is displayed using Plotly for interactive visualizations.

Libraries Used

- **Pandas**: For data manipulation and analysis.
- **Plotly Express**: For creating interactive visualizations.
- **Dash**: For building the web application.
- **JupyterDash**: To run Dash applications directly within Jupyter Notebooks.

Data Source

The dataset is loaded from the following URL:

- COVID-19 Data

App Layout

The app layout consists of:

1. **Header**: Displays the title of the dashboard.
2. **Dropdown Menu**: Allows users to select a country from the dataset.
3. **Date Range Picker**: Users can specify a date range for the data.
4. **Graphs**: Two line graphs that show:
   - Total COVID-19 cases over the selected date range.
   - Total COVID-19 deaths over the selected date range.

Code Explanation

Data Loading

The dataset is loaded directly from the URL using Pandas:

url = 'https://covid.ourworldindata.org/data/owid-covid-data.csv'

df = pd.read_csv(url)

App Initialization

The app is initialized using JupyterDash to allow it to run within the Jupyter Notebook: app = JupyterDash(**name**)

Layout Definition

The layout of the app is defined using HTML components and Dash Core Components (dcc): app.layout = html.Div(children=[...])

Callback Function

The callback function updates the graphs based on user input from the dropdown and date picker: @app.callback(...) def update_graphs(selected_country, start_date, end_date): ...

The function filters the DataFrame based on the selected country and date range, then creates the line graphs using Plotly Express.

Running the App

The app is run inline in the notebook with the following command: app.run_server(mode='inline', debug=True)

Future Enhancements

- Implement error handling for cases where no data is returned for the selected filters.
- Add more metrics, such as new cases or vaccination rates.
- Enhance visual styling using CSS or Dash Bootstrap Components.
- Include a download button for users to export the filtered data.
- Provide user instructions or tooltips to improve usability.

Conclusion

This COVID-19 dashboard serves as a practical example of using Dash and Plotly for data visualization. It allows users to interact with the data and gain insights into the COVID-19 situation across different countries and time periods.

In [1]:

```python
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import pandas as pd
import plotly.express as px

# Load the dataset directly from the URL
url = 'https://covid.ourworldindata.org/data/owid-covid-data.csv'
df = pd.read_csv(url)

# Initialize the Dash app
app = dash.Dash(__name__)

# Define the layout of the app
app.layout = html.Div(children=[
    html.H1(children='COVID-19 Dashboard', style={'textAlign': 'center', 'color': '#007BFF'}),

    # Dropdown for selecting the country
    dcc.Dropdown(
        id='country-dropdown',
        options=[{'label': country, 'value': country} for country in df['location'].unique()],
        value='United States',  # Default value
        multi=False,
        style={'marginBottom': '20px'}
    ),

    # Date Range Picker
    dcc.DatePickerRange(
        id='date-picker',
        start_date=df['date'].min(),
        end_date=df['date'].max(),
        display_format='YYYY-MM-DD',
        style={'marginBottom': '20px'}
    ),

    # Graph for displaying total cases
    dcc.Graph(id='cases-graph', style={'marginBottom': '20px'}),

    # Graph for displaying total deaths
    dcc.Graph(id='deaths-graph')
])

# Callback to update both graphs based on selected country and date range
@app.callback(
    Output('cases-graph', 'figure'),
    Output('deaths-graph', 'figure'),
    Input('country-dropdown', 'value'),
    Input('date-picker', 'start_date'),
    Input('date-picker', 'end_date')
)
def update_graphs(selected_country, start_date, end_date):
    # Filter the DataFrame based on user input
    filtered_df = df[(df['location'] == selected_country) &
                     (df['date'] >= start_date) &
                     (df['date'] <= end_date)]

    # Create total cases figure
    cases_fig = px.line(filtered_df, x='date', y='total_cases',
                        title=f'Total COVID-19 Cases in {selected_country}')

    # Create total deaths figure
    deaths_fig = px.line(filtered_df, x='date', y='total_deaths',
                         title=f'Total COVID-19 Deaths in {selected_country}')

    return cases_fig, deaths_fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```