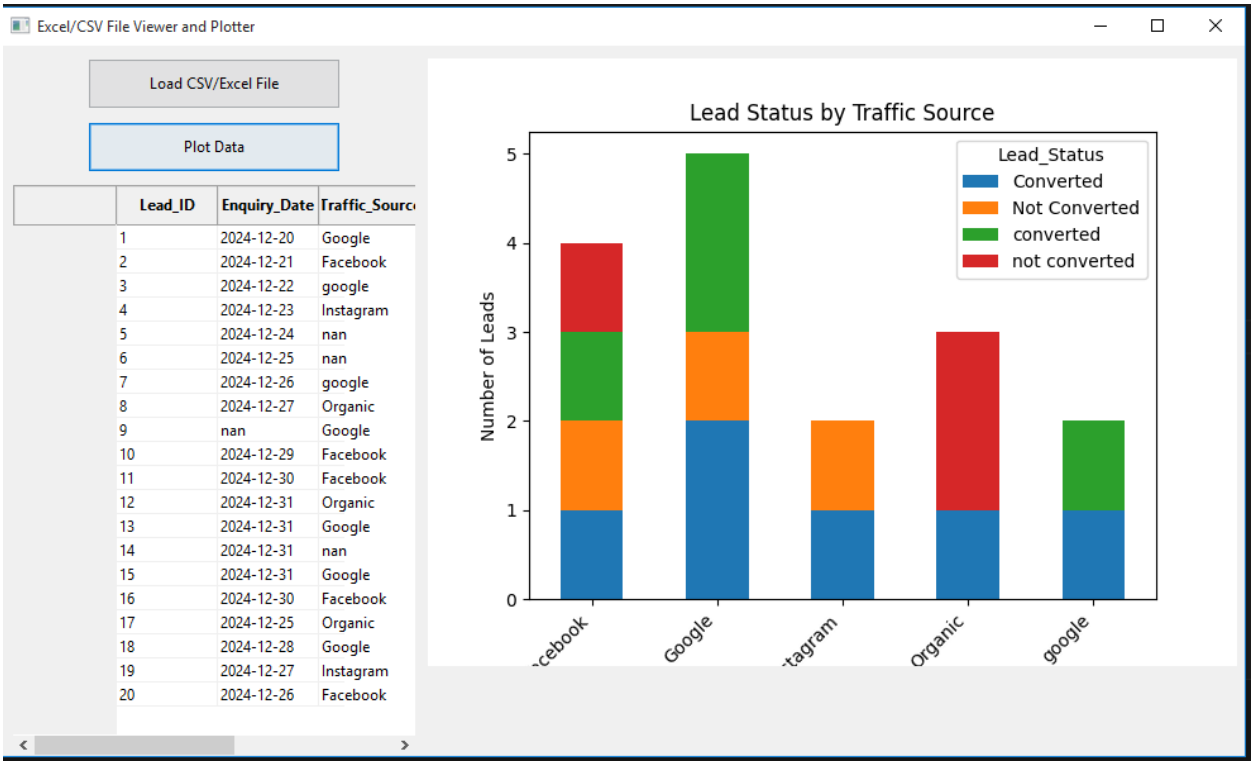
## Wxpython projects on simple data analysis on Csv file:-

| Lead_ID | Enquiry_Date | Traffic_Source | Lead_Status | Lead_Amount |
|---|---|---|---|---|
| 1 | 12/20/2024 | Google | Converted | 1500 |
| 2 | 12/21/2024 | Facebook | Not Converted | 200 |
| 3 | 12/22/2024 | google | converted | 500 |
| 4 | 12/23/2024 | Instagram | Not Converted | 300 |
| 5 | 12/24/2024 | | Converted | 1200 |
| 6 | 12/25/2024 | | not converted | 100 |
| 7 | 12/26/2024 | google | Converted | 350 |
| 8 | 12/27/2024 | Organic | not converted | 600 |
| 9 | | Google | Converted | 1000 |
| 10 | 12/29/2024 | Facebook | Converted | 2500 |
| 11 | 12/30/2024 | Facebook | | 500 |
| 12 | 12/31/2024 | Organic | Converted | 800 |
| 13 | 12/31/2024 | Google | converted | 700 |
| 14 | 12/31/2024 | | Converted | 600 |
| 15 | 12/31/2024 | Google | Not Converted | 150 |
| 16 | 12/30/2024 | Facebook | converted | 1200 |
| 17 | 12/25/2024 | Organic | not converted | 220 |
| 18 | 12/28/2024 | Google | converted | 1800 |
| 19 | 12/27/2024 | Instagram | Converted | 1000 |
| 20 | 12/26/2024 | Facebook | not converted | 3000 |



```
import wx
import pandas as pd
import wx.grid
import matplotlib.pyplot as plt
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigureCanvas

class FileDisplayFrame(wx.Frame):
    def __init__(self, *args, **kw):
        super(FileDisplayFrame, self).__init__(*args, **kw)
        self.InitUI()

    def InitUI(self):
        # Set the frame size
        self.SetSize((1000, 600))
        self.SetTitle('Excel/CSV File Viewer and Plotter Data Analysis')

        # Create panel
        panel = wx.Panel(self)

        # Horizontal sizer to hold grid and graph side by side
        hbox = wx.BoxSizer(wx.HORIZONTAL)

        # Vertical sizer for left side (grid)
        vbox_left = wx.BoxSizer(wx.VERTICAL)

        # Button to load the file
        self.btnLoadFile = wx.Button(panel, label='Load CSV/Excel File',
size=(200, 40))
        vbox_left.Add(self.btnLoadFile, flag=wx.ALIGN_CENTER|wx.TOP, border=10)

        # Button to plot the data
        self.btnPlotData = wx.Button(panel, label='Plot Data', size=(200, 40))
        vbox_left.Add(self.btnPlotData, flag=wx.ALIGN_CENTER|wx.TOP, border=10)

        # Grid to display the data
        self.grid = wx.grid.Grid(panel)
        vbox_left.Add(self.grid, 1, flag=wx.EXPAND|wx.TOP, border=10)

        # Add left side (grid) to the horizontal sizer
        hbox.Add(vbox_left, 1, flag=wx.EXPAND|wx.LEFT, border=10)

        # Vertical sizer for right side (plot)
        vbox_right = wx.BoxSizer(wx.VERTICAL)

        # wx.Panel for matplotlib figure canvas
```

```
        self.canvas_panel = wx.Panel(panel)
        self.fig, self.ax = plt.subplots()
        self.canvas = FigureCanvas(self.canvas_panel, -1, self.fig)
        vbox_right.Add(self.canvas_panel, 1, flag=wx.EXPAND|wx.TOP, border=10)

        # Add right side (plot) to the horizontal sizer
        hbox.Add(vbox_right, 2, flag=wx.EXPAND|wx.LEFT, border=10)

        # Bind button events
        self.btnLoadFile.Bind(wx.EVT_BUTTON, self.onLoadFile)
        self.btnPlotData.Bind(wx.EVT_BUTTON, self.onPlotData)

        # Set the panel sizer
        panel.SetSizer(hbox)

        self.data = None  # Store the loaded data

    def onLoadFile(self, event):
        # Open file dialog to choose CSV or Excel file
        with wx.FileDialog(self, "Open CSV or Excel file", wildcard="CSV files
(*.csv)|*.csv|Excel files (*.xlsx)|*.xlsx",
                           style=wx.FD_OPEN | wx.FD_FILE_MUST_EXIST) as
fileDialog:
            if fileDialog.ShowModal() == wx.ID_CANCEL:
                return  # User canceled file selection

            # Get the selected file path
            path = fileDialog.GetPath()

            try:
                # Load data from file (CSV or Excel)
                if path.endswith('.csv'):
                    self.data = pd.read_csv(path)
                elif path.endswith('.xlsx'):
                    self.data = pd.read_excel(path)
                else:
                    wx.MessageBox("Unsupported file format!", "Error",
wx.ICON_ERROR)
                    return

                # Display the data in the grid
                self.displayData(self.data)
            except Exception as e:
                wx.MessageBox(f"Failed to load file: {str(e)}", "Error",
wx.ICON_ERROR)
```

```
    def displayData(self, data):
        # Clear any previous data in the grid
        self.grid.ClearGrid()

        # Set number of rows and columns
        self.grid.CreateGrid(len(data), len(data.columns))

        # Populate grid with column names (headers)
        for col, col_name in enumerate(data.columns):
            self.grid.SetColLabelValue(col, col_name)

        # Populate grid with data values
        for row in range(len(data)):
            for col in range(len(data.columns)):
                self.grid.SetCellValue(row, col, str(data.iloc[row, col]))

    def onPlotData(self, event):
        if self.data is None:
            wx.MessageBox("No data loaded to plot.", "Error", wx.ICON_ERROR)
            return

        # Ensure the necessary columns exist
        if 'Traffic_Source' not in self.data.columns or 'Lead_Status' not in
self.data.columns:
            wx.MessageBox("Required columns 'Traffic_Source' and 'Lead_Status'
are not present.", "Error", wx.ICON_ERROR)
            return

        # Clear previous plot
        self.ax.clear()

        # Count occurrences of each Traffic_Source and Lead_Status combination
        grouped_data = self.data.groupby(['Traffic_Source',
'Lead_Status']).size().unstack(fill_value=0)

        # Debug: Print the grouped data
        print(grouped_data)

        # Plot the data as a bar chart
        grouped_data.plot(kind='bar', stacked=True, ax=self.ax)

        # Set the plot title and labels
        self.ax.set_title("Lead Status by Traffic Source")
        self.ax.set_xlabel('Traffic Source')
```

```
        self.ax.set_ylabel('Number of Leads')

        # Rotate x-axis labels for better readability
        self.ax.set_xticklabels(self.ax.get_xticklabels(), rotation=45,
ha="right")

        # Redraw the canvas
        self.canvas.draw()

def main():
    app = wx.App(False)
    frame = FileDisplayFrame(None)
    frame.Show()
    app.MainLoop()

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

## Overview

The code creates a **wxPython** application that allows the user to:

- Load a CSV or Excel file.
- Display the content of that file in a **grid**.
- Plot a **stacked bar chart** based on specific columns (`Traffic_Source` and `Lead_Status`).

It uses:

- **wxPython**: For building the graphical user interface (GUI).
- **Pandas**: To load and manipulate the data from CSV or Excel files.
- **Matplotlib**: To plot the data.

## Code Explanation

### *Import Statements*

```
python
Copy code
import wx
import pandas as pd
import wx.grid
import matplotlib.pyplot as plt
```

```
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as
FigureCanvas
```

- **wx**: Main package for creating the GUI.
- **pandas**: Used to load and process CSV/Excel data.
- **wx.grid**: Provides a grid control (similar to a spreadsheet) for displaying tabular data.
- **matplotlib.pyplot**: Used for plotting data (stacked bar chart).
- **FigureCanvasWxAgg**: Bridges the `matplotlib` plot with the `wxPython` GUI

### *`FileDisplayFrame` Class (Main Frame)*

This is the main window of the application where everything happens.

```
python
Copy code
class FileDisplayFrame(wx.Frame):
    def __init__(self, *args, **kw):
        super(FileDisplayFrame, self).__init__(*args, **kw)
        self.InitUI()
```

- **FileDisplayFrame**: This class inherits from `wx.Frame` and defines the main window of the application.
- The `__init__` method calls `InitUI()`, which initializes the user interface elements.

### *User Interface (UI) Initialization: `InitUI()`*

```
python
Copy code
def InitUI(self):
    self.SetSize((1000, 600))
    self.SetTitle('Excel/CSV File Viewer and Plotter Data Analysis')

    # Create panel
    panel = wx.Panel(self)
```

- Sets the window size and title.
- A `wx.Panel` is created inside the frame to hold all the widgets.

### *Horizontal Box Sizer: `hbox`*

```
python
Copy code
hbox = wx.BoxSizer(wx.HORIZONTAL)
```

- A `BoxSizer` is used to manage the layout of widgets. `wx.HORIZONTAL` indicates that widgets will be placed horizontally.

### *Left Side Layout (Grid and Buttons)*

```
python
Copy code
vbox_left = wx.BoxSizer(wx.VERTICAL)
```

```
self.btnLoadFile = wx.Button(panel, label='Load CSV/Excel File', size=(200,
40))
vbox_left.Add(self.btnLoadFile, flag=wx.ALIGN_CENTER|wx.TOP, border=10)
```

- Creates a vertical box sizer `vbox_left` that will contain the "Load File" button and the "Plot Data" button.
- **Button**: A `wx.Button` is created to allow users to load a CSV or Excel file.

self.btnPlotData = wx.Button(panel, label='Plot Data', size=(200, 40)) vbox_left.Add(self.btnPlotData, flag=wx.ALIGN_CENTER|wx.TOP, border=10)

- Another button for triggering the plot action.

### *Grid Setup: Displaying Data*

```
python
Copy code
self.grid = wx.grid.Grid(panel)
vbox_left.Add(self.grid, 1, flag=wx.EXPAND|wx.TOP, border=10)
```

- A `wx.grid.Grid` is created to display the contents of the CSV/Excel file.
- The `grid` is added to the `vbox_left` sizer. The flag `wx.EXPAND` ensures that the grid expands to fill available space.

### *Right Side Layout (Plotting Area)*

```
python
Copy code
vbox_right = wx.BoxSizer(wx.VERTICAL)
self.canvas_panel = wx.Panel(panel)
self.fig, self.ax = plt.subplots()
self.canvas = FigureCanvas(self.canvas_panel, -1, self.fig)
vbox_right.Add(self.canvas_panel, 1, flag=wx.EXPAND|wx.TOP, border=10)
```

- A vertical sizer `vbox_right` holds the plot.
- A `matplotlib` figure (`fig`) and axis (`ax`) are created using `plt.subplots()`.
- `FigureCanvasWxAgg` is used to embed the `matplotlib` plot inside the wxPython window.

### *Adding Sizers to the Main Layout*

```
python
Copy code
hbox.Add(vbox_left, 1, flag=wx.EXPAND|wx.LEFT, border=10)
hbox.Add(vbox_right, 2, flag=wx.EXPAND|wx.LEFT, border=10)
```

- The left side (`vbox_left`, containing buttons and grid) and right side (`vbox_right`, containing the plot) are added to the horizontal sizer `hbox`.

### *Setting the Panel's Layout*

```
python
```

```
Copy code
panel.SetSizer(hbox)
```

- This sets the layout for the entire `panel` by assigning `hbox` as the sizer.

### *Data Attribute*

```
python
Copy code
self.data = None  # Store the loaded data
```

- Initializes `self.data` to `None`. This attribute will store the loaded data (from the CSV or Excel file).

---

## Loading the File: `onLoadFile()`

```
python
Copy code
def onLoadFile(self, event):
    with wx.FileDialog(self, "Open CSV or Excel file", wildcard="CSV files
(*.csv)|*.csv|Excel files (*.xlsx)|*.xlsx",
                       style=wx.FD_OPEN | wx.FD_FILE_MUST_EXIST) as
fileDialog:
        if fileDialog.ShowModal() == wx.ID_CANCEL:
            return
```

- This method is triggered when the user clicks the "Load File" button.
- It opens a file dialog to let the user choose either a CSV or Excel file. The `wildcard` argument restricts the file selection to only CSV and Excel files.

```
python
Copy code
path = fileDialog.GetPath()
try:
    if path.endswith('.csv'):
        self.data = pd.read_csv(path)
    elif path.endswith('.xlsx'):
        self.data = pd.read_excel(path)
    else:
        wx.MessageBox("Unsupported file format!", "Error", wx.ICON_ERROR)
        return
```

- Once the user selects a file, the code reads the file into a `pandas` DataFrame using either `pd.read_csv` or `pd.read_excel`, depending on the file type.
- If the file format is unsupported, it shows an error message.

```
python
Copy code
self.displayData(self.data)
```

- The `displayData` method is called to populate the `wx.grid.Grid` with the loaded data.

---

## Displaying Data in the Grid: `displayData()`

```
python
Copy code
def displayData(self, data):
    self.grid.ClearGrid()
    self.grid.CreateGrid(len(data), len(data.columns))

    for col, col_name in enumerate(data.columns):
        self.grid.SetColLabelValue(col, col_name)

    for row in range(len(data)):
        for col in range(len(data.columns)):
            self.grid.SetCellValue(row, col, str(data.iloc[row, col]))
```

- Clears any previous data in the grid.
- Creates a grid with the number of rows and columns based on the `data` (loaded file).
- Populates the column headers (`SetColLabelValue`).
- Fills in each cell with the data values from the DataFrame (`SetCellValue`).

---

## Plotting Data: `onPlotData()`

```
python
Copy code
def onPlotData(self, event):
    if self.data is None:
        wx.MessageBox("No data loaded to plot.", "Error", wx.ICON_ERROR)
        return

    if 'Traffic_Source' not in self.data.columns or 'Lead_Status' not in
self.data.columns:
        wx.MessageBox("Required columns 'Traffic_Source' and 'Lead_Status'
are not present.", "Error", wx.ICON_ERROR)
        return
```

- This method is triggered when the "Plot Data" button is clicked.
- It checks if the data is loaded (`self.data` is not `None`) and if the required columns (`Traffic_Source` and `Lead_Status`) exist in the data.

```
python
Copy code
grouped_data = self.data.groupby(['Traffic_Source',
'Lead_Status']).size().unstack(fill_value=0)
```

- Groups the data by `Traffic_Source` and `Lead_Status` and counts the occurrences using `.size()`.
- The `unstack()` method reshapes the grouped data into a wide format (columns for each `Lead_Status` value).
- The `fill_value=0` ensures that missing combinations are filled with zeros.

```
python
Copy code
grouped_data.plot(kind='bar', stacked=True, ax=self.ax)
```

- Plots the grouped data as a **stacked bar chart** using `matplotlib`.

```
python
Copy code
self.ax.set_title("Lead Status by Traffic Source")
self.ax.set_xlabel('Traffic Source')
self.ax.set_ylabel('Number of Leads')
```

- Sets the title and labels for the x and y axes.

```
python
Copy code
self.ax.set_xticklabels(self.ax.get_xticklabels(), rotation=45, ha="right")
```

- Rotates the x-axis labels to make them readable if they are long.

```
python
Copy code
self.canvas.draw()
```

- Redraws the canvas to display the updated plot.

---

## Main Method: Running the App

```
python
Copy code
def main():
    app = wx.App(False)
    frame = FileDisplayFrame(None)
    frame.Show()
    app.MainLoop()
```

- Creates a `wx.App` (which runs the event loop for the application) and a `FileDisplayFrame` (the main window).
- The event loop is started with `app.MainLoop()`.

## Conclusion:

This program combines wxPython, pandas, and matplotlib to create an interactive application that can:

- Load data from CSV/Excel files.
- Display it in a grid.
- Plot a stacked bar chart based on the data.