GUI APPLICATION

In [ ]:

```

```

In [12]:

```python
#start of predefine code
import pandas as pd
from ipywidgets import Button, Dropdown, Output, VBox, Layout, widgets
from IPython.display import display
from IPython.display import clear_output
from tkinter import Tk, filedialog
import matplotlib.pyplot as plt
import numpy as np

graph_type = ['Choose one.. ','bubble','bar']
funtionality = ['Choose One','Sort','Filter']
sort_option = ['ascending','descending']
df = ''
new_df = ''
input_box = ''
input_fontsize = ''
input_title = ''
#end of predefine code

def select_files(b):
    # Clear the previous output on the output cell.
    clear_output()

    # Declare df as global.
    global df

    # Create and hide the root window of the tkinter library.
    root = Tk()
    root.withdraw()

    # Open the file from the file dialog and store the file name in a variable.
    file_name = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")])

    # Read the data from the selected CSV file and store the data in the df variabl
    df = pd.read_csv(file_name)

    # Create a dropdown widget for selecting the sorting or filtering the data from
    functionality = ['Choose One', 'Sort', 'Filter']
    function_widget = widgets.Dropdown(options=functionality)

    # Define a variable function_int and store the drop down widget which is linked
    function_int = widgets.interactive(choose_the_function, function=function_widge

    # Display the dropdown.
    display(function_int)

    def sort_dataframe(column, type_of_sort, head_range):
        global new_df, df
        try:
            print(df[column].dtypes)
            if df[column].dtypes != 'float' and df[column].dtypes != 'int':
                df[column] = df[column].astype(float)
            if type_of_sort == 'ascending':
                new_df = df.sort_values(by=column, ascending=True)
                display(new_df.head(head_range))
            else:
                new_df = df.sort_values(by=column, ascending=False)
                display(new_df.head(head_range))
        except:
            print('The data is not structured so cannot perform the selected action

        def filter_dataframe(filter_column, comparison, head_df):
            global new_df, df, input_box

            if comparison == "=":
                new_df = df[df[filter_column] == input_box.value]
                new_df = new_df.head(head_df)
                display(new_df)
            elif comparison == ">":
                new_df = df[df[filter_column] > input_box.value]
                new_df = new_df.head(head_df)
                display(new_df)
            elif comparison == "<":
                new_df = df[df[filter_column] < input_box.value]
                new_df = new_df.head(head_df)
                display(new_df)
            else:
                print('Choose correct option')

            get_widget()

        def get_widget():
            global new_df, input_title, input_fontsize

            # Create dropdown widget for x-axis label
            xlabel_widget = widgets.Dropdown(options=df.columns, description='X-axi

            # Create dropdown widget for y-axis label
            ylabel_widget = widgets.Dropdown(options=df.columns, description='Y-axi

            # Create input box for graph title
            input_title = widgets.Text(description='Title')
            input_fontsize = widgets.Text(description='Font size:')

            # Display the input box for graph title
            display(input_title)

            # Display the input box for font size
            display(input_fontsize)

            # Create dropdown widget for graph type
            graph_type = ['Choose one.. ', 'bubble', 'bar']
            graph_widget = widgets.Dropdown(options=graph_type, description='Graph

            # Define a variable to store all dropdown widgets and bind them to the
            graph = widgets.interactive(display_plot,
                                        xaxis=xlabel_widget,
                                        yaxis=ylabel_widget,
                                        graph_type=graph_widget)

            # Display all the widgets inside the graph variable
            display(graph)

    # Bar graph plot
    def plot_bar_graph(x_col, y_col):
        fig = px.bar(df, x=x_col, y=y_col, color=y_col)
        fig.show()

# Bubble graph plot
    def plot_bubble_graph(x_col, y_col, size_col):
        fig = px.scatter(df, x=x_col, y=y_col, size=size_col, color=size_col, hover
        fig.show()


        def choose_the_function(function):
            global sort_option, df, input_box

            if function == 'Sort':
                sort_col_widget = widgets.Dropdown(options=df.columns)
                sort_option_widget = widgets.Dropdown(options=sort_option)
                range_widget = widgets.Dropdown(options=[20,30,40])

                sort_int = widgets.interactive(sort_dataframe,
                                               column=sort_col_widget,
                                               type_of_sort=sort_option_widget,
                                               head_range=range_widget)
                display(sort_col_widget, sort_option_widget, range_widget)

            elif function == 'Filter':
                display(df)
                input_box = widgets.Text(description="Value:")
                display(input_box)

                filter_col_widget = widgets.Dropdown(options=df.columns)
                compare_widget = widgets.Dropdown(options=['Choose the option..', '
                head_widget = widgets.Dropdown(options=[20,30,40])

                groupby_int = widgets.interactive(filter_dataframe,
                                                  filter_column=filter_col_widget,
                                                  Comparison=compare_widget,
                                                  value=input_box,
                                                  head_df=head_widget)
                display(filter_col_widget, compare_widget, head_widget)




#start of predefine code
def display_plot(xaxis, yaxis, graph_type):
    global new_df
    global input_title
    global input_fontsize
    if(graph_type == 'bubble'):
        plt.subplots(figsize=(19,8))
        rgb = np.random.rand(3)
        #Write Condition here
        if new_df[yaxis].min() > 1000:
            plt.scatter(new_df[xaxis], new_df[yaxis], c=rgb, alpha=0.4, s=new_df[ya
        elif new_df[yaxis].min() < 100:
            plt.scatter(new_df[xaxis], new_df[yaxis], c=rgb, alpha=0.4, s=new_df[ya
        else:
            plt.scatter(new_df[xaxis], new_df[yaxis], c=rgb, alpha=0.4, s=new_df[ya
        #End of write condition here
        plt.title(input_title.value, fontsize=input_fontsize.value)
        plt.xlabel(xaxis, fontsize=input_fontsize.value)
        plt.xticks(rotation='vertical')
        plt.ylabel(yaxis, fontsize=input_fontsize.value)
        plt.show()
    elif(graph_type == 'bar'):
        plt.subplots(figsize=(19,8))
        plt.bar(new_df[xaxis], new_df[yaxis], color=['red', 'green','blue','yellow'
        plt.title(input_title.value, fontsize=input_fontsize.value)
        plt.xlabel(xaxis, fontsize=input_fontsize.value)
        plt.xticks(rotation='vertical')
        plt.ylabel(yaxis, fontsize=input_fontsize.value)
        plt.show()
    else:
        print("Choose valid graph")
fileselect = widgets.Button(description="File select")
fileselect.on_click(select_files)
display(fileselect)
#end of predefined

```

File select

In [ ]:

```

```