Course: DATS 6401 – Visualization of Complex Data

Instructor: Dr. Reza Jafari

Lab Number: 5

Lab Name: App using Dash

Name: Rehapriadarsini Manikandasamy

RM	04.06.22
Initials	Date

Abstract

The lab focuses on deploying a dash app which accommodates multiple tabs on GCP. The app includes accessing CONVENIENT_global_confirmed_cases.csv dataset and creating different dash core components throughout the assignment. Plots of quadratic functions and polynomial orders were generated using dynamic Inputs from users. The calculator tab helps in performing basic math operations. The lab finally generated a link to the app deployed on GCP with all the above defined functionalities.

CHAPTER 1

INTRODUCTION

The lab helps to understand how to load data and display as an interactive web base application using Python. The dashboard includes different tabs each containing outputs for the given assignment questions. The final app will be deployed on GCP to be accessed across the different users or any device.

The tab1 includes visualizing the confirmed COVID19 cases of different countries. The dropdown menu allows to select countries and graph will be updated based on the selection. The tab2 involves plotting a quadratic equation $f(x) = ax2 + bx + c$ with sliders to select the variable values. The tab3 will be helpful in performing basic arithmetic operations of the given two numbers.

The tab4 helps in visualizing the histogram of a gaussian distribution with the mean, standard deviation, sample size and bins dynamically chosen by user using the sliders. The tab5 is to plot the equation based on the provided polynomial order. The last tab involves visualizing the subplots of a created data frame with a slider component.

The app will be deployed on GCP using the docker and enabling api services to get deployed on the web. The proper deployment of app will result in generation of web link which can be accessed by any user.

A detailed explanation of methods and packages are explained in the next chapter.

CHAPTER 2

METHOD, THEORY AND PROCEDURES

Dash Tabs:

The dcc.Tabs and dcc.Tab components can be used to create tabbed sections in your app. The dcc.Tab component controls the style and value of the individual tab and the dcc.Tabs component hold a collection of dcc.Tab components.

Dash Callbacks:

functions that are automatically called by Dash whenever an input component's property changes, to update some property in another component (the output).

Dash Core Components:

The Dash Core Components module (dash.dcc) can be imported and used with from dash import dcc and gives you access to many interactive components, including, dropdowns, checklists, and sliders.

Dash HTML Components:

Dash is a web application framework that provides pure Python abstraction around HTML, CSS, and JavaScript. Instead of writing HTML or using an HTML templating engine, you compose your layout using Python with the Dash HTML Components module (dash.html).

Procedure:

1. Load the necessary libraries and dataset
2. Do necessary preprocessing required
3. Create dash layouts with layouts for each tab individually
4. Apply callback functions to each tab
5. On the GCP console, create a docker and python file to include the necessary codes for working
6. Perform the instructions mentioned on the GCP deployment PDF to deploy the app on web
7. Access the generated link to use the app in future

CHAPTER 3

ANSWERS TO ASKED QUESTIONS

Required libraries:

The following python libraries are required to run the lab code file seamlessly.

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

1. Using Dash in python write an app that plot the COVID global confirmed cased (the same dataset as in LAB # 3) for the following countries: a. US b. Brazil c. United Kingdom_sun d. China_sum e. India f. Italy g. Germany Hint: You need to develop a dropdown menu with the list of countries. Make sure to add a title to your app. Use the external style sheet as follow. Add the title to the drop dropdown menu as: “Pick the country Name”. external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']
   Answer:

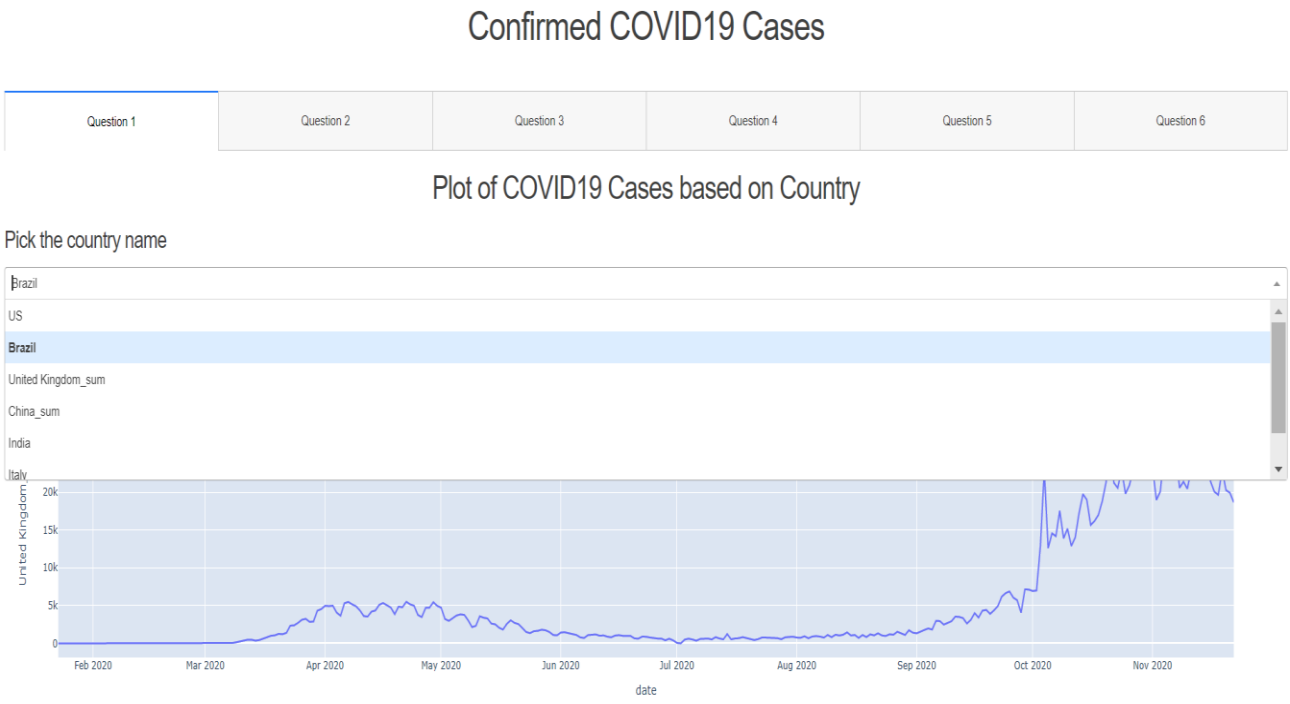

Fig 1: Output for Question 1

2. Create an app using Dash that plots the quadratic function $f(x) = ax2 + bx + c$ for x between -2, 2 with 1000 samples. The a, b and c must be an input with a slider component. Add an appropriate title and label to each section of your app. Use the same external style sheet as question 1.
   Answer:

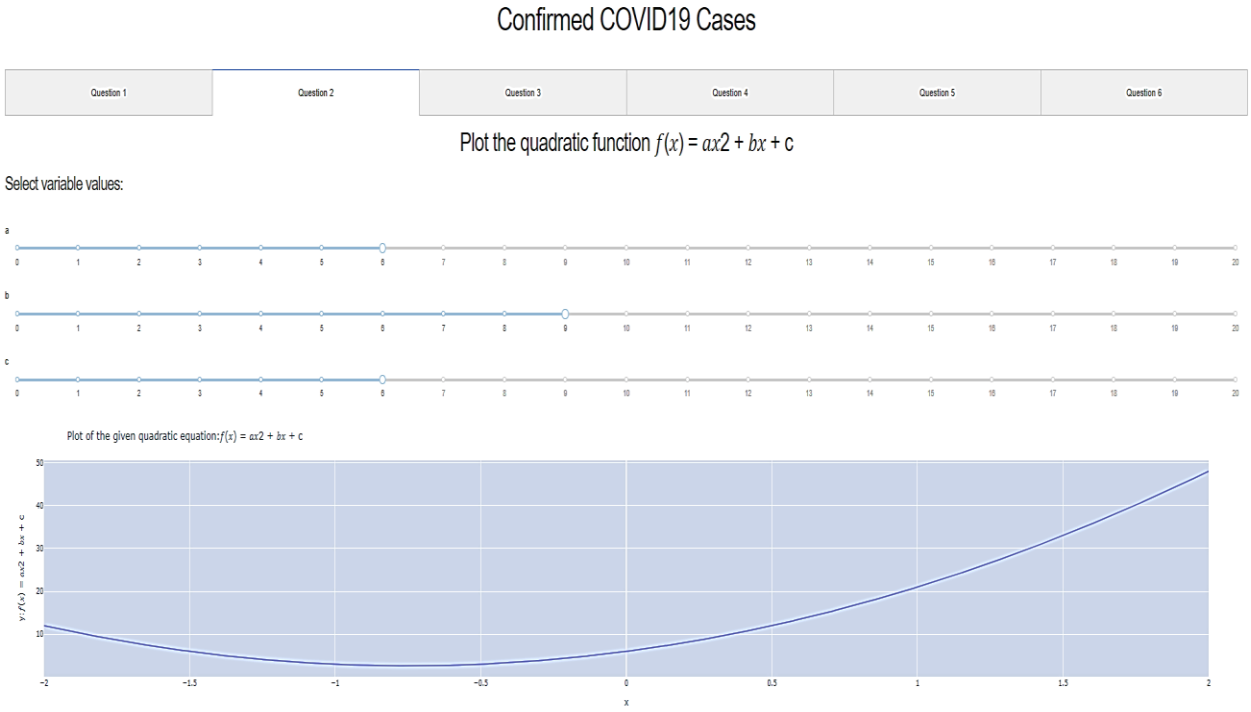

Fig 2: Output for Question 2

3. Create a calculator app using Dash that perform the following basic math operations:
   Answer:

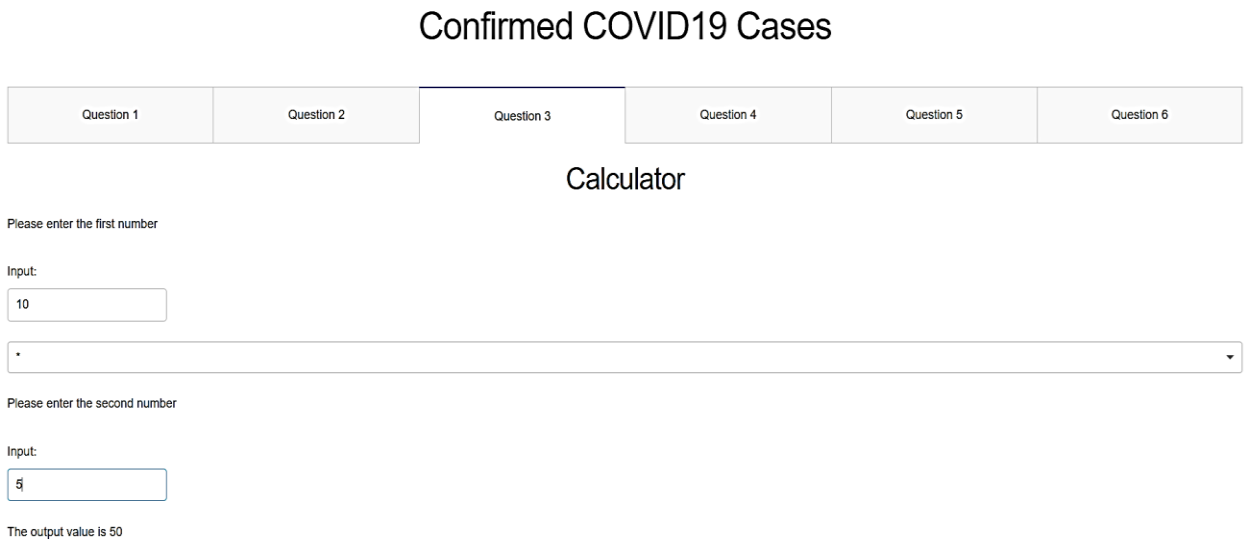

Fig 3: Output for Question 3

4. Develop an interactive web-based app using Dash in python that plot the histogram plot for the gaussian distribution. The mean, std, number of samples and number of bins must be entered through a separate slider bar (one slider for each). The mean ranges between -2, 2 with step size of 1. The std ranges from 1 to 3 with step size of 1. Number of samples ranges from 100-10000 with the step size of 500. The number of bins ranges from 20 to 100 with step size of 10.

   Answer:

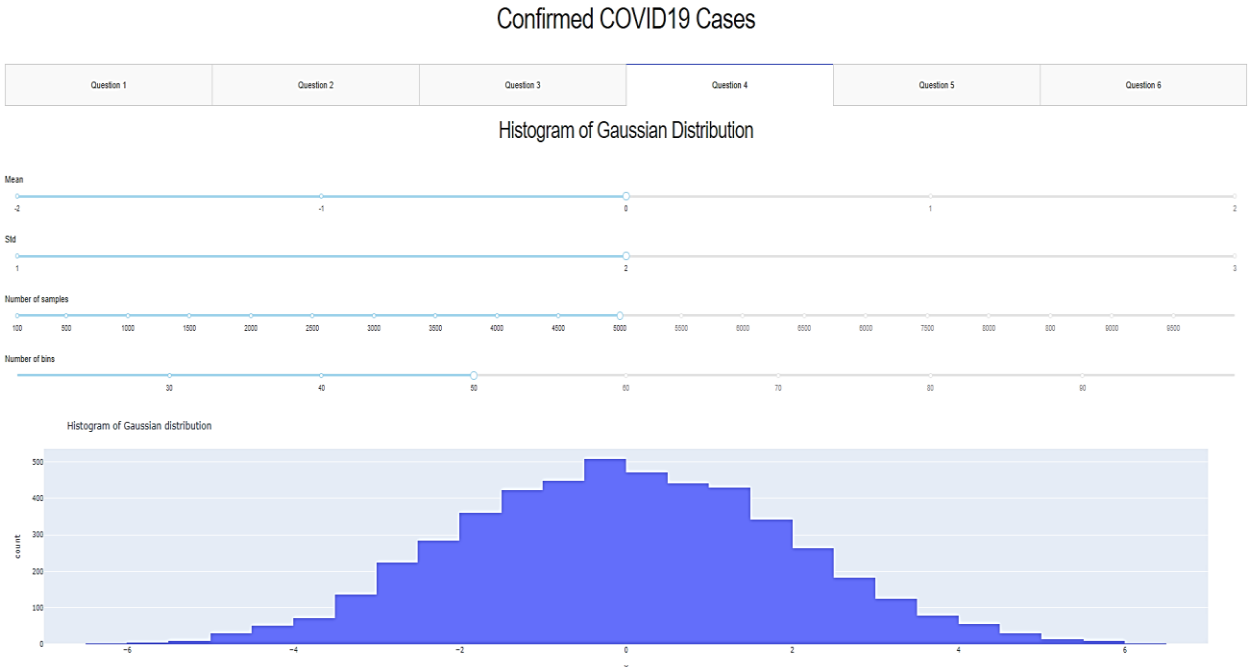

Fig 4: Output for Question 4

5. Develop an interactive web-based app using Dash in python that plot polynomial function by entering the order of the polynomial through an input field. For example, if the input entered number to input field is 2, then the function $f(x) = x2$ must be plotted. If the entered number to the input field is 3, then the function $f(x) = x3$ must be plotted. The range of x is -2, 2 with 1000 samples in between.
   Answer:

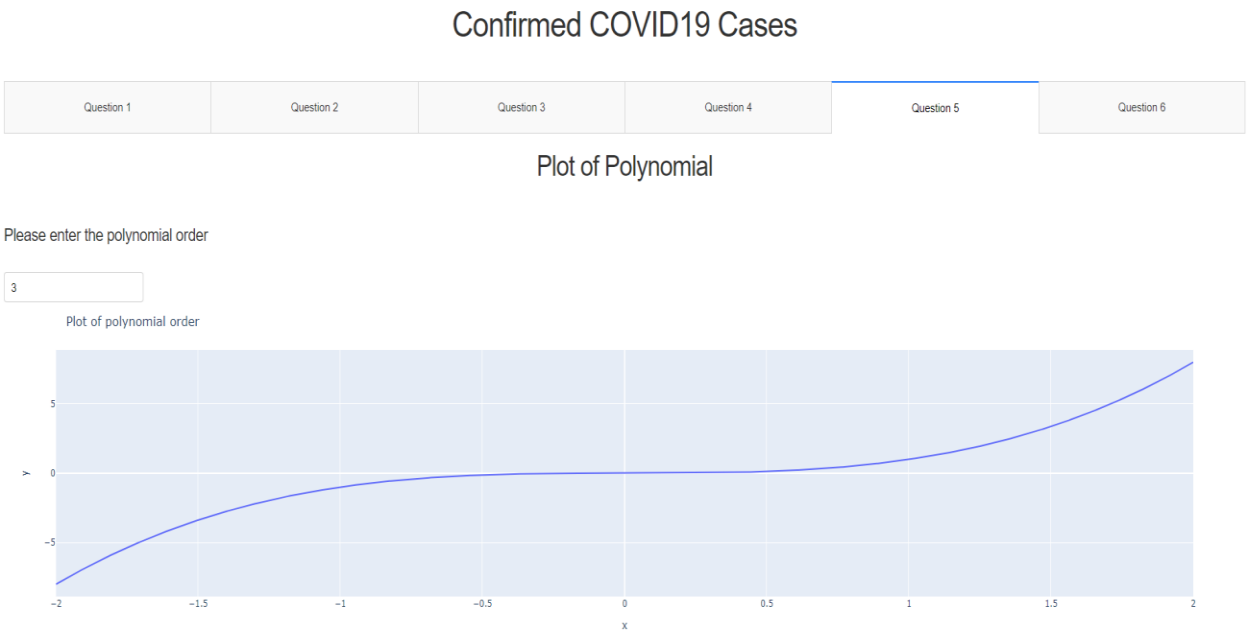

Fig 5: Output for Question 5

6. Create a dashboard that displays the group bar plot 2x2. Bellow each plot place a slider with min=0 and max = 20 and step size = 1. Add the label with H1 for ‘Hello Dash’ and H5 for the ‘Slider number’ label. The final web-based app should look like bellow

   Answer:

Fig 6: Output for Question 6

GCP Deployment link:

The above app has been deployed on the GCP. The app can be accessed at https://dashapp-75ln4rvlva-ue.a.run.app/

(Note: Outputs are little blurred since they were captured by adjusting the browser window size to screenshot the complete output. The app can be viewed in a better quality using the above link)

CONCLUSION

The primary objective of this lab is to generate an interactive web-based application which accommodates different tabs. The tab on the app helps in visualizing a pandas data frame, a quadratic equation, a basic calculator, and subplots. The app will be containing plots like line, histograms, and bars. The final app was deployed on GCP, and the app can be accessed at https://dashapp-75ln4rvlva-ue.a.run.app/

APPENDIX

Lab5_RM.py

```
import dash as dash
import numpy as np
from dash import dcc
from dash import html
import pandas as pd
import math

import plotly.express as px
from dash.dependencies import Input,Output

external_stylesheets=['https://codepen.io/chriddyp/pen/bWLwgP.css']
df=pd.read_csv('https://raw.githubusercontent.com/rjafari979/Complex-Data-
Visualization-/main/CONVENIENT_global_confirmed_cases.csv')
df=df.dropna(axis=0,how='any')

col_name=[]
df['China_sum']=df.iloc[0:,57:90].astype(float).sum(axis=1)
df['United Kingdom_sum']=df.iloc[0:,249:260].astype(float).sum(axis=1)

for col in df.columns:
    col_name.append(col)

df_covid=df[col_name]
df_covid['date']=pd.date_range(start='01-23-20',end='11-22-20')
df_bar = pd.DataFrame({
 "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
 "Amount": [4, 1, 2, 2, 4, 5],
 "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
})

my_app=dash.Dash('My_app',external_stylesheets=external_stylesheets)


my_app.layout=html.Div([html.H1('Confirmed COVID19
Cases',style={'textAlign':'center'}),
                        html.Br(),
                        dcc.Tabs(id='lab-qn',
                                 children=[
                                     dcc.Tab(label='Question 1',value='q1'),
                                     dcc.Tab(label='Question 2',value='q2'),
                                     dcc.Tab(label='Question 3',value='q3'),
                                     dcc.Tab(label='Question 4',value='q4'),
                                     dcc.Tab(label='Question 5',value='q5'),
                                     dcc.Tab(label='Question 6',value='q6'),
                                 ]),
                        html.Div(id='layout')

])

q1_layout=html.Div([
```

```
    html.H2('Plot of COVID19 Cases based on
Country',style={'textAlign':'center'}),
    html.H4('Pick the country name'),
   dcc.Dropdown(id='drop1',
                options=[
                    {'label':'US','value':'US'},
                    {'label': 'Brazil', 'value': 'Brazil'},
                    {'label': 'United Kingdom_sum', 'value': 'United
Kingdom_sum'},
                    {'label': 'China_sum', 'value': 'China_sum'},
                    {'label': 'India', 'value': 'India'},
                    {'label': 'Italy', 'value': 'Italy'},
                    {'label': 'Germany', 'value': 'Germany'},

                ],value='US',clearable=False),
    html.Br(),
    dcc.Graph(id='graphcovid')
])

q2_layout=html.Div([
    html.H2('Plot the quadratic function f(x) = ax2 + bx +
c',style={'textAlign':'center'}),
    html.H4('Select variable values:'),
    html.Br(),
    html.P('a'),
    dcc.Slider(id='slide1',min=0,max=20,step=1,value=1),
    html.Br(),
    html.P('b'),
    dcc.Slider(id='slide2',min=0,max=20,step=1,value=1),
    html.Br(),
    html.P('c'),
    dcc.Slider(id='slide3',min=0,max=20,step=1,value=1),
    html.Br(),
    dcc.Graph(id='graphquad')
])

q3_layout=html.Div([
    html.H2('Calculator',style={'textAlign':'center'}),
    html.P('Please enter the first number'),
    html.Br(),
    html.P('Input:',style={'display':'inline-block','margin-right':10}),
    html.Br(),

dcc.Input(id='input1',type='text',placeholder='',style={'display':'inline-
block'}),
    html.Br(),html.Br(),
    dcc.Dropdown(id='drop2',
                options=[
                    {'label':'+','value':'+'},
                    {'label': '-', 'value': '-'},
                    {'label': '*', 'value': '*'},
                    {'label': '/', 'value': '/'},
                    {'label': 'log', 'value': 'log'},
                    {'label': 'square', 'value': 'square'},
                    {'label': 'Root square', 'value': 'Root square'}
                ],value='US',clearable=False),
    html.Br(),
```

```
    html.P('Please enter the second number'),
    html.Br(),
    html.P('Input:',style={'display':'inline-block','margin-right':10}),
    html.Br(),

dcc.Input(id='input2',type='text',placeholder='',style={'display':'inline-
block'}),
    html.Br(),html.Br(),
    html.Div(id='output3')

])

q4_layout=html.Div([
    html.H2('Histogram of Gaussian
Distribution',style={'textAlign':'center'}),
    html.Br(),
    html.P('Mean'),
    dcc.Slider(id='mean',min=-2,max=2,value=0,
               marks={-2:'-2',-1:'-1',0:'0',1:'1',2:'2'}),
    html.Br(),
    html.P('Std'),
    dcc.Slider(id='std',min=1,max=3,value=1,
               marks={1:'1',2:'2',3:'3'}),
    html.Br(),
    html.P('Number of samples'),
    dcc.Slider(id='samples',min=100,max=10000,value=100,

marks={100:'100',500:'500',1000:'1000',1500:'1500',2000:'2000',

2500:'2500',3000:'3000',3500:'3500',4000:'4000',4500:'4500',5000:'5000',

5500:'5500',6000:'6000',6500:'6500',7000:'6000',7500:'7500',8000:'8000',8500:
'800',9000:'9000',9500:'9500'}),
    html.Br(),
    html.P('Number of bins'),
    dcc.Slider(id='bins',min=20,max=100,value=20,

marks={30:'30',40:'40',50:'50',60:'60',70:'70',80:'80',90:'90'}),
    html.Br(),
    dcc.Graph(id='graph_hist')

])

q5_layout = html.Div([
    html.H2('Plot of Polynomial',style={'textAlign':'center'}),
    html.Br(),
    html.H5('Please enter the polynomial order'),
    html.Br(),
    dcc.Input(id='poly',type='number'),
    html.Br(),
    dcc.Graph(id='graph_poly')
])
fig1=px.bar(data_frame=df_bar,x='Fruit',y='Amount',color='City',barmode='grou
p')
fig2 = px.bar(data_frame=df_bar, x='Fruit', y='Amount',
color='City',barmode='group')
fig3 = px.bar(data_frame=df_bar, x='Fruit', y='Amount',
```

```
color='City',barmode='group')
fig4 = px.bar(data_frame=df_bar, x='Fruit', y='Amount',
color='City',barmode='group')
q6_layout=html.Div([
    html.Div(
        children=[
            html.H1('Hello Dash1'),
            html.P('Dash:A web application framework in python.'),
            dcc.Graph(figure=fig1),
            dcc.Slider(id='dash1',min=0,max=20,step=1,value=10),
            html.Br(),
            html.H1('Hello Dash3'),
            html.P('Dash:A web application framework in python.'),
            dcc.Graph(figure=fig3),
            dcc.Slider(id='dash3',min=0,max=20,step=1,value=10)
        ],style={'padding': 10, 'flex': 1}
    ),
html.Div(
        children=[
            html.H1('Hello Dash2'),
            html.P('Dash:A web application framework in python.'),
            dcc.Graph(figure=fig2),
            dcc.Slider(id='dash2',min=0,max=20,step=1,value=10),
            html.Br(),
            html.H1('Hello Dash4'),
            html.P('Dash:A web application framework in python.'),
            dcc.Graph(figure=fig4),
            dcc.Slider(id='dash4',min=0,max=20,step=1,value=10)
        ],style={'padding': 10, 'flex': 1}
    ),
],style={'display': 'flex', 'flex-direction': 'row'})


@my_app.callback(Output(component_id='layout',component_property='children'),
                 [Input(component_id='lab-qn',component_property='value')
                ])



def update_layout(ques):
    if ques=='q1':
        return q1_layout
    elif ques=='q2':
        return q2_layout
    elif ques=='q3':
        return q3_layout
    elif ques=='q4':
        return q4_layout
    elif ques=='q5':
        return q5_layout
    elif ques=='q6':
        return q6_layout

@my_app.callback(Output(component_id='graphcovid',component_property='figure'
),
                 [Input(component_id='drop1',component_property='value')])
def update_graph(input):
```

```
    fig = px.line(df_covid,x='date',y=input)
    fig.update_layout(title='Plot of COVID19 Cases based on Country')
    return fig


@my_app.callback(Output(component_id='graphquad',component_property='figure')
,
                [Input(component_id='slide1',component_property='value'),
                 Input(component_id='slide2',component_property='value'),
                 Input(component_id='slide3',component_property='value')])

def update_quad(slide1,slide2,slide3):
    x=np.linspace(-2,2,1000)
    y=(slide1*x**2)+(slide2*x)+slide3
    fig=px.line(x=x,y=y,labels={'y':'y:f(x) = ax2 + bx + c'})
    fig.update_layout(title='Plot of the given quadratic equation:f(x) = ax2 +
bx + c')
    return fig

@my_app.callback(Output(component_id='output3',component_property='children')
,
                 [Input(component_id='input1',component_property='value'),
                  Input(component_id='drop2',component_property='value'),
                  Input(component_id='input2',component_property='value')])
def update_calc(a,op,b):
    a=int(a)
    b=int(b)
    if op=='+':
        return f'The output value is {a+b}'
    elif op=='-':
        return f'The output value is {a - b}'
    elif op=='*':
        return f'The output value is {a * b}'
    elif op=='/':
        return f'The output value is {a / b}'
    elif op=='log':
        return f'The output value is {math.log(a,b)}'
    elif op=='square':
        return f'The output value is : a squared ={a**2},b squared={b**2}'
    elif op=='Root square':
        return f'The output value is: a= {math.sqrt(a)},b={math.sqrt(b)}'

@my_app.callback(Output(component_id='graph_hist',component_property='figure'
),
    [Input(component_id='mean',component_property='value'),
     Input(component_id='std',component_property='value'),
Input(component_id='samples',component_property='value'),
Input(component_id='bins',component_property='value')

     ]
)
def display_color(mean,std,samples,bins):
    x=np.random.normal(mean,std,size=samples)
    fig=px.histogram(x=x,nbins=bins,range_x=[-7,7])
    fig.update_layout(title='Histogram of Gaussian distribution')
    return fig
```

```
@my_app.callback(Output(component_id='graph_poly',component_property='figure'
),
                 [Input(component_id='poly',component_property='value')])
def update_poly(inp):
    x=np.linspace(-2,2,1000)
    y=x**inp
    fig=px.line(x=x,y=y)
    fig.update_layout(title='Plot of polynomial order')
    return fig


my_app.run_server(port=8009,host='0.0.0.0')
```

REFERENCES

- https://community.plotly.com/t/how-to-manage-the-layout-of-division-figures-in-dash/6484/2
- https://dash.plotly.com/dash-core-components/tabs