4%20-%20Dashboard%20Walkthrough

April 12, 2023

Topics

1 Just plotly

[ ]:
```
# Step 1 - Just a bar chart - No dashboard

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math


ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob.head()
```

[ ]:
```
obm = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
↪reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
↪reset_index()

fig = px.bar(obm, x="Month", y="Illnesses")
fig.show()
```

2 Building a Dashboard

2.1 Dashboard Components

- html components - https://dash.plotly.com/dash-html-components
- core components - https://dash.plotly.com/dash-core-components

2.2 The general structure

[ ]:
```
# The general structure of a dashboard application:

imports .....

app = JupyterDash(__name__)      # This is the start of the application

get the data....

create a figure(plot)...

app.layout =                     # Describe what the page will look like

    layout code

    dcc.Graph()                 # What plot will be included

@app.callback(
    what are the inputs?
    what are the outputs?

    resusable component )       # This processes the input and creates the␣
 ↪output

app.run_server(mode='inline')   # .run_server() is the method to run the code
```

2.3 Required imports

[ ]:
```
#!pip install jupyter-dash
```

[ ]:
```
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html
```

[ ]:
```
ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
obm = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
```

3 Dashboard with an empty figure - dcc.graph() & html.Div()

[ ]:
```
# Create an empty figure here
fig = go.Figure()

app = JupyterDash(__name__)

# Layout the dashboard

# app.layout - html.Div( something goes in the .Div )
# app.layout - html.Div( [sometimes a list of things go into the .Div] )
# app.layout - html.Div( [sometimes other .Divs go in the .Div] )

app.layout = html.Div([        # passing in a list of 'things' to Div
    html.H1('Hello Jim'),      # This line generates <h1>Hello Jim</h1>
    html.Div('''
        An Empty Dashboard
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
])
#app.run_server(mode='inline')
app.run_server(mode='external', port = 8090)
```

4 Adding a plot to the figure

[ ]:
```
# STEP 3 - Add the bar chart to the dashboard

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math



ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# Create a bar chart here.  Use x="Month", y="Illnesses"
fig = px.bar(ob_month, x="Month", y="Illnesses")

app = JupyterDash(__name__)

app.layout = html.Div([       # passing in a list of 'things' to Div
    html.H1('Hello Jim'),     # This line generates <h1>Hello Jim</h1>
    html.H3('''
        Foodborne Illness Outbreaks
    '''),

    dcc.Graph(
        id='illnesses-graph',
        figure=fig
    )
])
#app.run_server(mode='inline')
app.run_server(mode='external', port = 8091)  #This will only work if you are␣
 ↪running from your local machine.
```

5 Change plot to a scatterplot

```
[ ]: # point is.. nothing else needs to change!
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html                              # Has a component for every HTML tag

import pandas as pd
import plotly.express as px

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig = px.scatter(obs, x='Illnesses', y='Hospitalizations',
                 size='Fatalities', color='State', hover_name='State',
                 size_max=60)

app = JupyterDash(__name__)

app.layout = html.Div([       # passing in a list of 'things' to Div
    html.H1('Hello Jim'),     # This line generates <h1>Hello Jim</h1>
    html.Div('''
        Foodborne Illness Outbreaks
    '''),
    dcc.Graph(
        id='ill-vs-hosp',
        figure=fig
    ) ])

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8092)
```

6 Add multiple plots

[ ]:
```
# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig1 = px.scatter(obs,
                  x='Illnesses',
                  y='Hospitalizations',
                  size='Fatalities',
                  color='State', )

fig2 = px.bar(ob_month, x="Month", y="Illnesses")

fig3 = px.bar(ob_month, x="Month", y="Hospitalizations")

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    dcc.Graph(id='x', figure=fig1 ),
    dcc.Graph(id='y', figure=fig2 ),
    dcc.Graph(id='z', figure=fig3 )
])

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8093)
```

7 Modify the plot layout with html components

7.1 update_layout

[ ]:
```python
# STEP 4 - Modify the plot layout.

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math

app = JupyterDash(__name__)

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()


fig = px.bar(ob_month, x="Month", y="Illnesses")

colors = {
    'background': '#111111', # black
    'text': '#7FDBFF'        # light blue
}

fig.update_layout(
    plot_bgcolor=colors['background'],
    paper_bgcolor=colors['background'],
    font_color=colors['text']
)


app.layout = html.Div([       # passing in a list of 'things' to Div
    html.H1('Hello Jim'),     # This line generates <h1>Hello Jim</h1>

    html.Div('''
        Changing the background.
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
])
#app.run_server(mode='inline')
app.run_server(mode='external', port = 8094)
```

8 Modify the html component parameters

[ ]:
```
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math

app = JupyterDash(__name__)

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# STEP 1 - Change the Div style
             # style={'backgroundColor': colors['background']},children=  - - -␣
 ↪- - - - - -
# STEP 2 - Change style of first div
             # style={'textAlign': 'center','color': colors['text']}      - - -␣
 ↪- - - - - -
# STEP 3 - Change style of second div
             # style={'textAlign': 'center','color': colors['text']}      - - -␣
 ↪- - - - - -
```

```
fig = px.bar(ob_month, x="Month", y="Illnesses")

colors = {
    'background': '#111111', # black
    'text': '#7FDBFF'        # light blue
}


#fig = go.Figure()

app.layout = html.Div(style={'backgroundColor': colors['background']},children=[
    html.H1('Hello CDC', style={'textAlign': 'center','color': colors['text'] ␣
 ↪}),

    html.Div('Foodborne Illnesses by Month', style={'textAlign':␣
 ↪'center','color': colors['text']}),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
])
#app.run_server(mode='inline')
app.run_server(mode='external', port = 8095)
```

9 Using .Div to add multiple plots

[ ]:
```
# Adding multiple graphs to the dashboard

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
```

```
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig1 = px.scatter(obs, x='Illnesses', y='Hospitalizations',
                 size='Fatalities', color='State', hover_name='State',
                 size_max=60)
fig2 = px.bar(ob_month, x="Month", y="Illnesses")
fig3 = px.bar(ob_month, x="Month", y="Hospitalizations")

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    html.Div([dcc.Graph(id='x', figure=fig1 )]),
    html.Div([dcc.Graph(id='y', figure=fig2 )]),
    html.Div([dcc.Graph(id='z', figure=fig3 )])
])

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8096)
```

10 Positioning plots

[ ]:
```
# Adjusting the graph positions

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
```

```
# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig1 = px.scatter(obs, x='Illnesses', y='Hospitalizations',
                 size='Fatalities', color='State', hover_name='State',
                 size_max=60)
fig2 = px.bar(ob_month, x="Month", y="Illnesses")
fig3 = px.bar(ob_month, x="Month", y="Hospitalizations")

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    html.Div([dcc.Graph(id='x', figure=fig1 )],style={'width': '49%', 'display':
 ↪ 'inline-block', 'padding': '0 20'}),
    html.Div([dcc.Graph(id='y', figure=fig2 )],style={'display':␣
 ↪'inline-block', 'width': '49%'}),
    html.Div([dcc.Graph(id='z', figure=fig3 )],style={'display':␣
 ↪'inline-block', 'width': '49%'})
])


#app.run_server(mode='inline')
app.run_server(mode='external', port = 8097)
```

11 Repositioning plots with parameters

[ ]:
```
# Using multiple .Div

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html                          # Has a component for every HTML tag

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
```

```
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig1 = px.scatter(obs, x='Illnesses', y='Hospitalizations',
                 size='Fatalities', color='State', hover_name='State',
                 size_max=60)
fig2 = px.bar(ob_month, x="Month", y="Illnesses")
fig3 = px.bar(ob_month, x="Month", y="Hospitalizations")

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    html.Div([dcc.Graph(id='x', figure=fig1 )],style={'width': '49%', 'display':
 ↪ 'inline-block', 'padding': '0 20'}),
    html.Div([dcc.Graph(id='y', figure=fig2 ),
              dcc.Graph(id='z', figure=fig3 )],style={'display':␣
 ↪'inline-block', 'width': '49%'}),
])

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8098)
```

12 Dash Exercise 2 - 20 minutes

- Use the Diabetes Analysis Dashboard notebook.
- Add a new cell that will contain all of the dashboard code.
- Add dashboard code to show the scatterplot created earlier.
- Can you add two more of your created graphs?

13 Dash Core Components

13.1 Adding markdown text

[ ]:
```
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html                        # Has a component for every HTML tag
import plotly.express as px
import pandas as pd
```

```python
ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

# Create a scatterplot with x - illness, y = hospitalizations, color = state
#                            size = fatalities and hover = state

fig = px.scatter(obs, x='Illnesses', y='Hospitalizations',
                 size='Fatalities', color='State', hover_name='State',
                 size_max=60)

markdown_text = '''
### Dash and Markdown

This chart shows foodborne illnesses driving hospitalizations.
The data covers the years 1998 - 2015.  Individual observations
are aggregated to the month level across years
'''

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    dcc.Graph(
        id='ill-vs-hosp',
        figure=fig
    ),
    dcc.Markdown(markdown_text)
]
)

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8099)
```

13.2 Dropdowns, Sliders and Other Components

https://dash.plotly.com/dash-core-components

[ ]:
```python
# Assumes all imports, data manipulation, etc. is complete

# STEP 1 - Hard-coded dropdown
# STEP 2 - df generated dropdown
# STEP 3 - Add a hard-coded slider
# STEP 4 - text input

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html                          # Has a component for every HTML tag
import plotly.express as px
import pandas as pd

app = JupyterDash(__name__)

# Add dcc.Graph code here
app.layout = html.Div([
    dcc.Dropdown(
        options=[
            {'label': 'New York City', 'value': 'NYC'},
            {'label': 'Montréal', 'value': 'MTL'},
            {'label': 'San Francisco', 'value': 'SF'}
        ],
        value='MTL'
    ),

    dcc.Dropdown(id='dropdown', options=[
        {'label': i, 'value': i} for i in obs.State.unique()
    ], multi=True, placeholder='Filter by state...'),

    dcc.Slider(
        min=-5,
        max=10,
        step=0.5,
        value=-3
    ),

#     dcc.Slider(
#         min=0,
#         max=9,
#         marks={i: 'Label{}'.format(i) for i in oby.Year.unique()}
#     )

    dcc.Input(
        placeholder='Enter a value...',
        type='text',
        value=''
    )
]
)

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8100)
```

14 Quick Exercise - 10 minutes

Add a few more components to the code above.

15 Dash Callbacks

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

16 Create a resuable component

[ ]:
```
# Create a reusable component
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html                              # Has a component for every HTML tag

import pandas as pd

app = JupyterDash(__name__)

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')

def generate_table(ob, max_rows=10):
    return html.Table([
        html.Thead(
            html.Tr([html.Th(col) for col in ob.columns])
        ),
        html.Tbody([
            html.Tr([
                html.Td(ob.iloc[i][col]) for col in ob.columns
            ]) for i in range(min(len(ob), max_rows))
        ])
    ])

app.layout = html.Div([
    html.H4(children='Foodborne Illness Outbreaks'),
    generate_table(ob)
])


#app.run_server(mode='inline')
app.run_server(mode='external', port = 8101)
```

16.1 A simple callback interactive app

We've seen this before!

[ ]:
```
# An example of a callback from documentation
# Just changes the text that appears - no plotting

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import dcc
from dash import html

app = JupyterDash(__name__)

app.layout = html.Div([
    html.H6("Change the value in the text box to see callbacks in action!"),
    html.Div([
        "Input: ",
        dcc.Input(id='my-input', value='initial value', type='text')
    ]),
    html.Br(),
    html.Div(id='my-output'),

])


@app.callback(
    Output(component_id='my-output', component_property='children'),
    Input(component_id='my-input', component_property='value')
)
def update_output_div(input_value):
    return 'Output: {}'.format(input_value)

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8102)
```

[ ]:
```
# A very basic dashboard with a slider

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math

app = JupyterDash(__name__)

ob = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪d29f529308d4e8332491341fed135dc9cc5ca0df/outbreaks-dashboard.csv')
ob = ob.loc[ob['State']!= 'Multistate']
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()

df = ob.groupby(['Year','State'])[['Illnesses','Hospitalizations',␣
 ↪'Fatalities']].sum().reset_index()




# STEP 1 - Comment out existing dcc.Graph
# STEP 2 - Add new graph and slider code
# STEP 3 - Add @app.callback
# STEP 4 - Add user-defined function - update_scatter

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),

    dcc.RadioItems(id='yearselected',
                                 options=[2000,2001,2002],
                                 value=2000)
])


@app.callback(
    Output('graph-with-slider', 'figure'),
    Input('yearselected', 'value'))

def update_figure(xyz):
    filtered_df = df[df.Year == xyz]

    abc = px.scatter(filtered_df, x='Illnesses', y='Hospitalizations',
                     size='Fatalities', color='State', hover_name='State',
                     size_max=60)

    abc.update_layout(transition_duration=500)

    return abc


#app.run_server(mode='inline')
app.run_server(mode='external', port = 8103)
```

[ ]:
```
df.info()
```

[ ]:
```
    dcc.Slider(
        id='year-slider',
        min=df['Year'].min(),
        max=df['Year'].max(),
        value=df['Year'].min(),
        marks={str(year): str(year) for year in df['Year'].unique()},
        step=None
    ),
```

[ ]:
```
# Changing the variables to include in the plot

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math

app = JupyterDash(__name__)

df = pd.read_csv('https://plotly.github.io/datasets/country_indicators.csv')

available_indicators = df['Indicator Name'].unique()

app.layout = html.Div([
    html.Div([

        html.Div([
            dcc.Dropdown(
                id='xaxis-column',
                options=[{'label': i, 'value': i} for i in␣
↪available_indicators],
                value='Fertility rate, total (births per woman)'
            ),
            dcc.RadioItems(
                id='xaxis-type',
                options=[{'label': i, 'value': i} for i in ['Linear', 'Log']],
                value='Linear',
                labelStyle={'display': 'inline-block'}
            )
        ], style={'width': '48%', 'display': 'inline-block'}),

        html.Div([
            dcc.Dropdown(
                id='yaxis-column',
                options=[{'label': i, 'value': i} for i in␣
↪available_indicators],
                value='Life expectancy at birth, total (years)'
            ),
            dcc.RadioItems(
                id='yaxis-type',
                options=[{'label': i, 'value': i} for i in ['Linear', 'Log']],
                value='Linear',
                labelStyle={'display': 'inline-block'}
            )
        ], style={'width': '48%', 'float': 'right', 'display': 'inline-block'})
    ]),

    dcc.Graph(id='indicator-graphic'),

    dcc.Slider(
        id='year--slider',
        min=df['Year'].min(),
        max=df['Year'].max(),
        value=df['Year'].max(),
        marks={str(year): str(year) for year in df['Year'].unique()},
        step=None
    )
])


@app.callback(
    Output('indicator-graphic', 'figure'),
    Input('xaxis-column', 'value'),
    Input('yaxis-column', 'value'),
    Input('xaxis-type', 'value'),
    Input('yaxis-type', 'value'),
    Input('year--slider', 'value'))
def update_graph(xaxis_column_name, yaxis_column_name, xaxis_type, yaxis_type,␣
 ↪year_value):
    dff = df[df['Year'] == year_value]

    fig = px.scatter(x=dff[dff['Indicator Name'] == xaxis_column_name]['Value'],
                     y=dff[dff['Indicator Name'] == yaxis_column_name]['Value'],
                     hover_name=dff[dff['Indicator Name'] ==␣
 ↪yaxis_column_name]['Country Name'])

    fig.update_layout(margin={'l': 40, 'b': 40, 't': 10, 'r': 0},␣
 ↪hovermode='closest')

    fig.update_xaxes(title=xaxis_column_name,
                     type='linear' if xaxis_type == 'Linear' else 'log')

    fig.update_yaxes(title=yaxis_column_name,
                     type='linear' if yaxis_type == 'Linear' else 'log')

    return fig



#app.run_server(mode='inline')
app.run_server(mode='external', port = 8104)
```

17 Dash Exercise 3 - 30 minutes

- modify your Diatebetes Dashboard.
- Use the code above (as an example) to have dropdown list that change the data in the scatter plot.