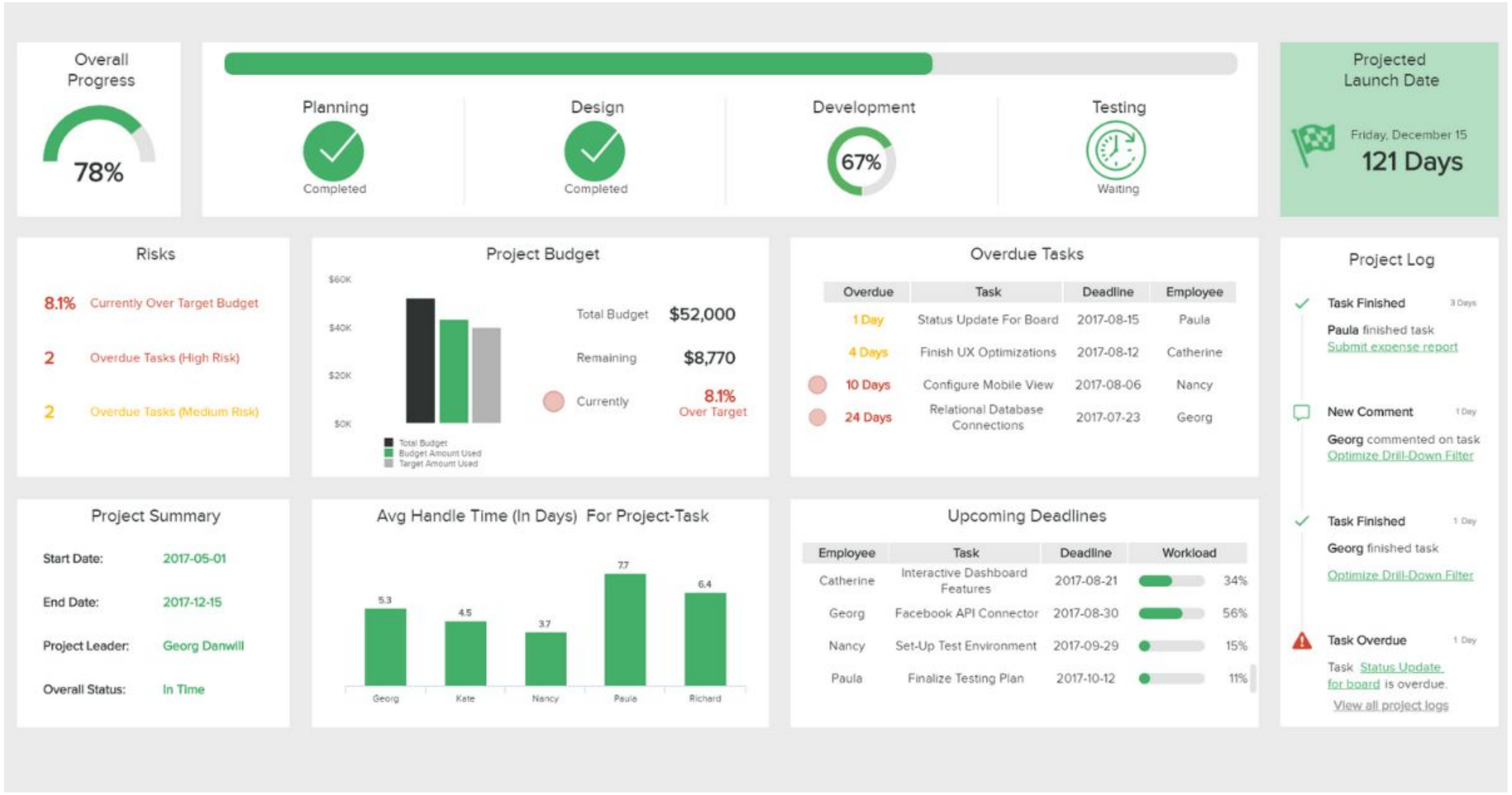
Data Visualization

Lecture 5

Building a Dashboard in Plotly Dash

What do we mean by a dashboard?

- IT Dashboard:



Overall Progress

78%

Planning — Completed

Design — Completed

Development — 67%

Testing — Waiting

Projected Launch Date

Friday, December 15

121 Days

Risks

8.1%	Currently Over Target Budget
2	Overdue Tasks (High Risk)
2	Overdue Tasks (Medium Risk)

Project Budget

Total Budget	$52,000
Remaining	$8,770
Currently	8.1% Over Target

Overdue Tasks

Overdue	Task	Deadline	Employee
1 Day	Status Update For Board	2017-08-15	Paula
4 Days	Finish UX Optimizations	2017-08-12	Catherine
10 Days	Configure Mobile View	2017-08-06	Nancy
24 Days	Relational Database Connections	2017-07-23	Georg

Project Log

Task Finished 3 Days
Paula finished task Submit expense report

New Comment 1 Day
Georg commented on task Optimize Drill-Down Filter

Task Finished 1 Day
Georg finished task Optimize Drill-Down Filter

Task Overdue 1 Day
Task Status Update for board is overdue.

View all project logs

Project Summary

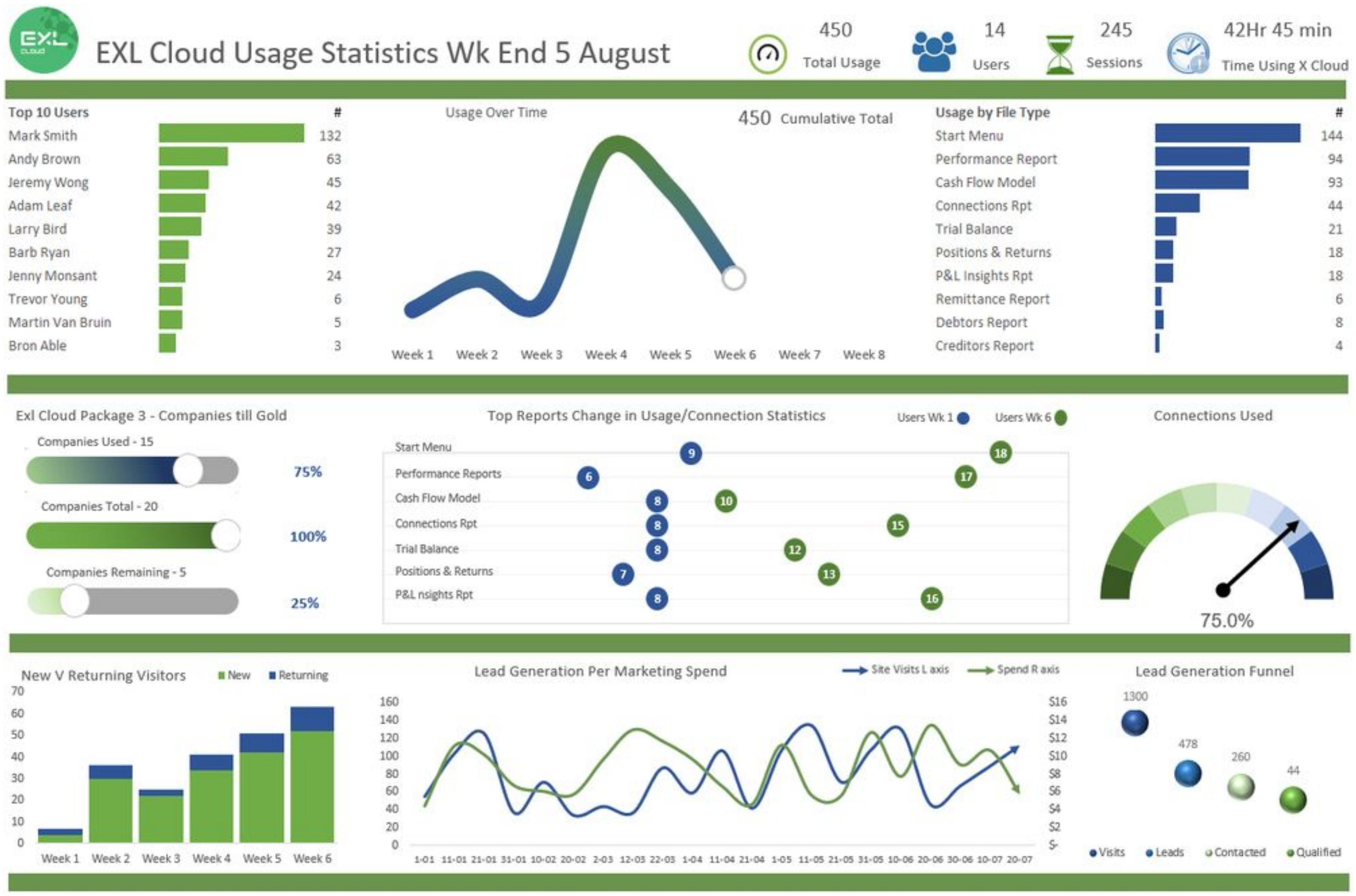
Start Date:	2017-05-01
End Date:	2017-12-15
Project Leader:	Georg Danwill
Overall Status:	In Time

Avg Handle Time (In Days) For Project-Task

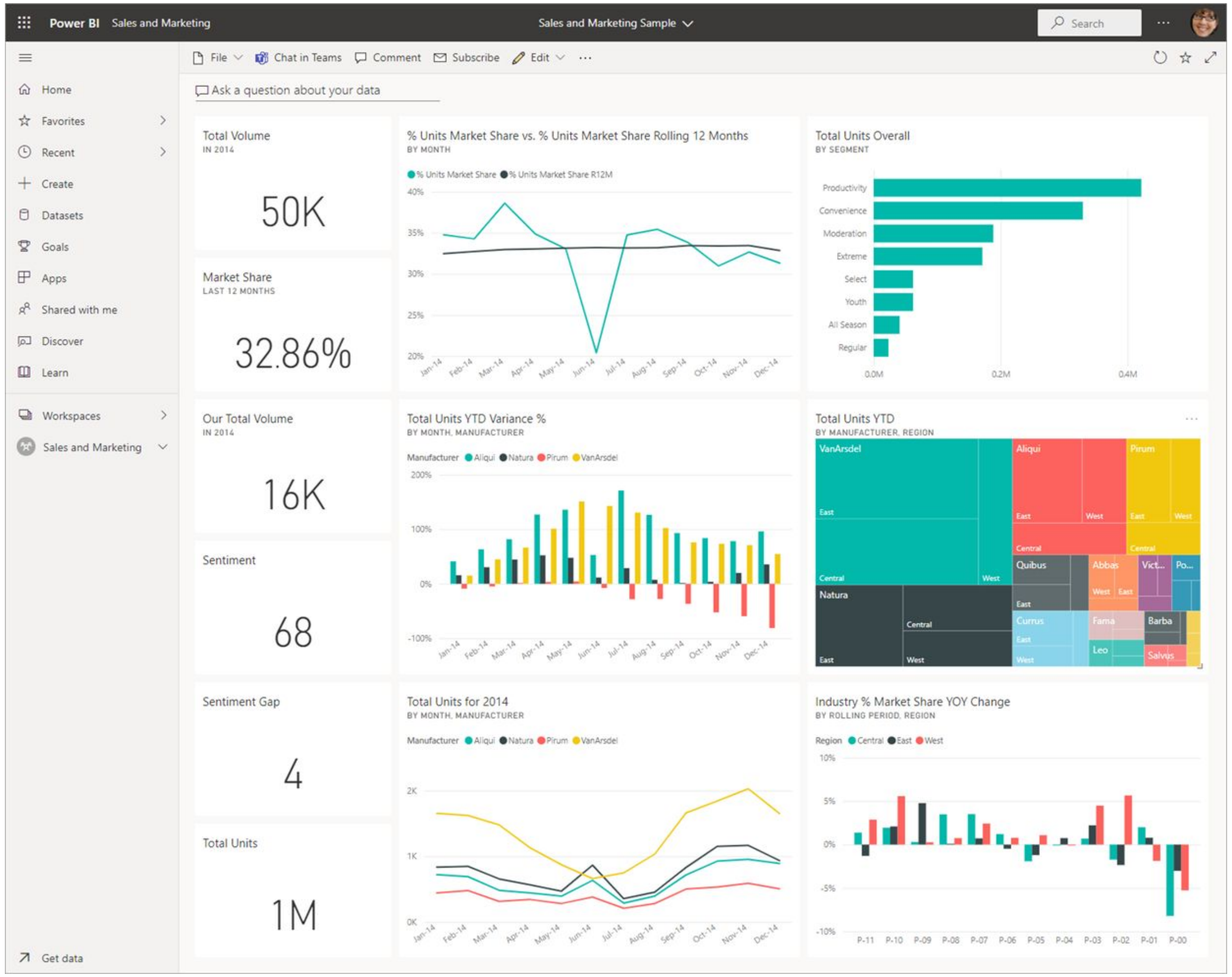
Upcoming Deadlines

Employee	Task	Deadline	Workload
Catherine	Interactive Dashboard Features	2017-08-21	34%
Georg	Facebook API Connector	2017-08-30	56%
Nancy	Set-Up Test Environment	2017-09-29	15%
Paula	Finalize Testing Plan	2017-10-12	11%

What do we mean by a dashboard?



What do we mean by a dashboard?


Power BI
Sales and Marketing
Sales and Marketing Sample
Search
File
Chat in Teams
Comment
Subscribe
Edit
Home
Favorites
Recent
Create
Datasets
Goals
Apps
Shared with me
Discover
Learn
Workspaces
Sales and Marketing
Get data
Ask a question about your data
Total Volume IN 2014 50K
Market Share LAST 12 MONTHS 32.86%
Our Total Volume IN 2014 16K
Sentiment 68
Sentiment Gap 4
Total Units 1M
% Units Market Share vs. % Units Market Share Rolling 12 Months BY MONTH
Total Units Overall BY SEGMENT
Productivity
Convenience
Moderation
Extreme
Select
Youth
All Season
Regular
Total Units YTD Variance % BY MONTH, MANUFACTURER
Manufacturer ● Aliqui ● Natura ● Pirum ● VanArsdel
Total Units YTD BY MANUFACTURER, REGION
VanArsdel
Aliqui
Pirum
Natura
Quibus
Abbas
Currus
Fama
Barba
Leo
Salvus
Total Units for 2014 BY MONTH, MANUFACTURER
Industry % Market Share YOY Change BY ROLLING PERIOD, REGION
Region ● Central ● East ● West

Why dashboards are useful?

- Admin dashboards are a great way to analyze data quickly and effectively.

- They collect and provide real-time data from your website or software to help make informed decisions for the organization.

Plotly Dash use cases

- Data visualization:
  - Allows users to create graphs and plots in a dashboard format to visualize large quantities of data, which would be impossible to analyze in its raw format.
- E-commerce:
  - Users can create responsive dashboards to analyze e-commerce trends and user behavior, including customer purchase patterns, customer churn patterns, payment-related information, etc.
- Sales dashboards:
  - Help organizations gain real-time insights into various sales categories (e.g., sales by region, date, time, etc). These insights help sales teams make informed decisions for the organization.

What is Dash?

- Dash is a python framework created by plotly for creating interactive web applications.

- Dash is written on the top of Flask, Plotly.js and React.js.

- With Dash, you don’t have to learn HTML, CSS and Javascript in order to create interactive dashboards, you only need python.

- Dash is open source and the application build using this framework are viewed on the web browser.

Building blocks of Dash

- Dash applications are made up of 2 building blocks :

1. Layout
   - Layout describes the look and feel of the app, it defines the elements such as graphs, dropdowns etc and the placement, size, color etc of these elements.

2. Callbacks
   - Callbacks are used to bring interactivity to the dash applications. These are the functions using which, for example, we can define the activity that would happen on clicking a button or a dropdown.

Installing needed libraries

- Use these commands on Jupyter notebook:
  - !pip install dash
  - !pip install dash-html-components
  - !pip install dash-core-components
  - !pip install dash_bootstrap_components
  - !pip install dash_bootstrap_templates

Importing libraries..

```
import dash
from dash import html
from dash import dcc
import plotly.graph_objects as go
import plotly.express as px
```

Dash Application

1. Initialize dash app using dash package.
2. Read the stock prices data for different companies from 2018 to 2019.

```
app = dash.Dash()   #initialising dash app
df = px.data.stocks() #reading stock price dataset
```

Dash Application

3. Create stock_prices function that returns the line chart for Google's stock prices:

```
def stock_prices():
    # Function for creating line chart showing Google stock prices over time
    fig = go.Figure([go.Scatter(x = df['date'], y = df['GOOG'],\
                     line = dict(color = 'firebrick', width = 4), name = 'Google')
                     ])
    fig.update_layout(title = 'Prices over time',
                      xaxis_title = 'Dates',
                      yaxis_title = 'Prices'
                      )
    return fig
```

Dash Application

4. Setting layout using HTML Div component:

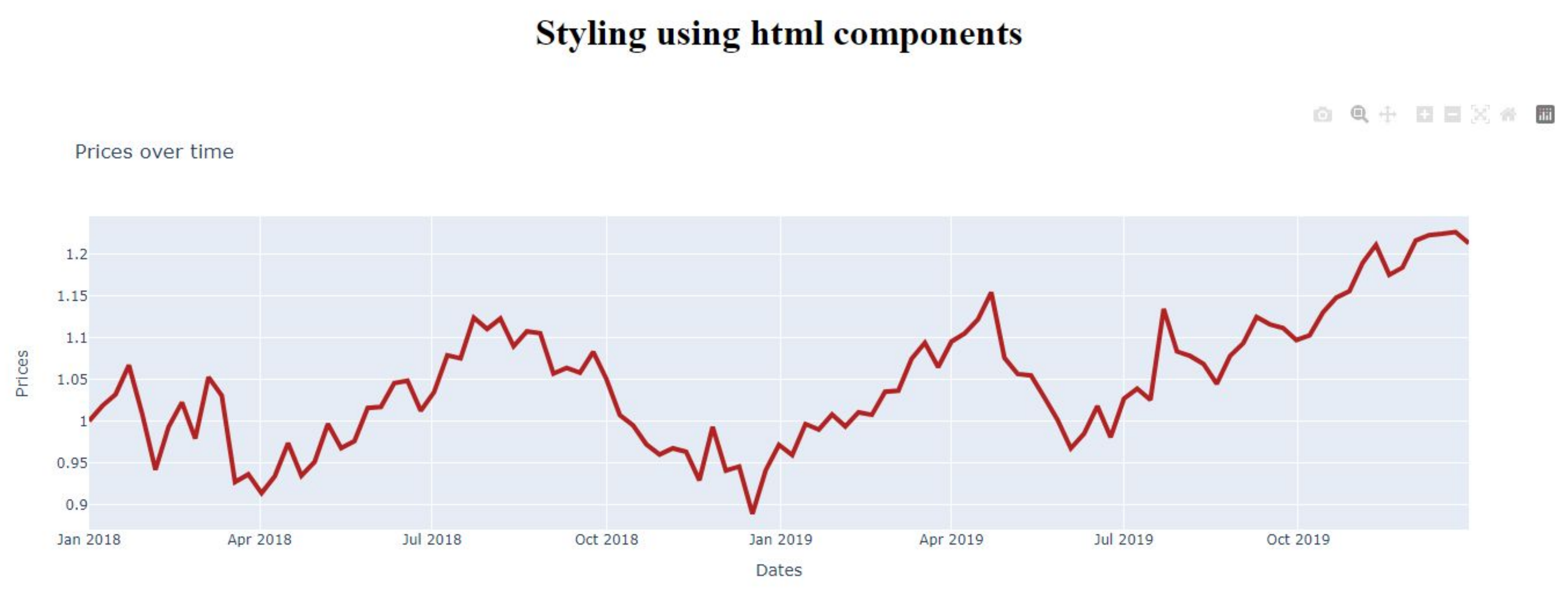
```
app.layout = html.Div(id = 'parent', children = [
   html.H1(id = 'H1', children = 'Styling using html components', style =
{'textAlign':'center’,\'marginTop':40,'marginBottom':40}),
      dcc.Graph(id = 'line_plot', figure = stock_prices())
   ]
            )
```

Dash Application

5. In order to view our application, we need to run our web server just like in Flask (Remember Dash is built on top of Flask):

```
if __name__ == '__main__':
    app.run()
```

Dash Application


Styling using html components

Dash Application

6. Create dropdown to select the stocks of Apple, Google or Amazon:

```
dcc.Dropdown( id = 'dropdown',
options = [
   {'label':'Google', 'value':'GOOG' },
   {'label': 'Apple', 'value':'AAPL'},
   {'label': 'Amazon', 'value':'AMZN'},
   ],
value = 'GOOG'
)
```

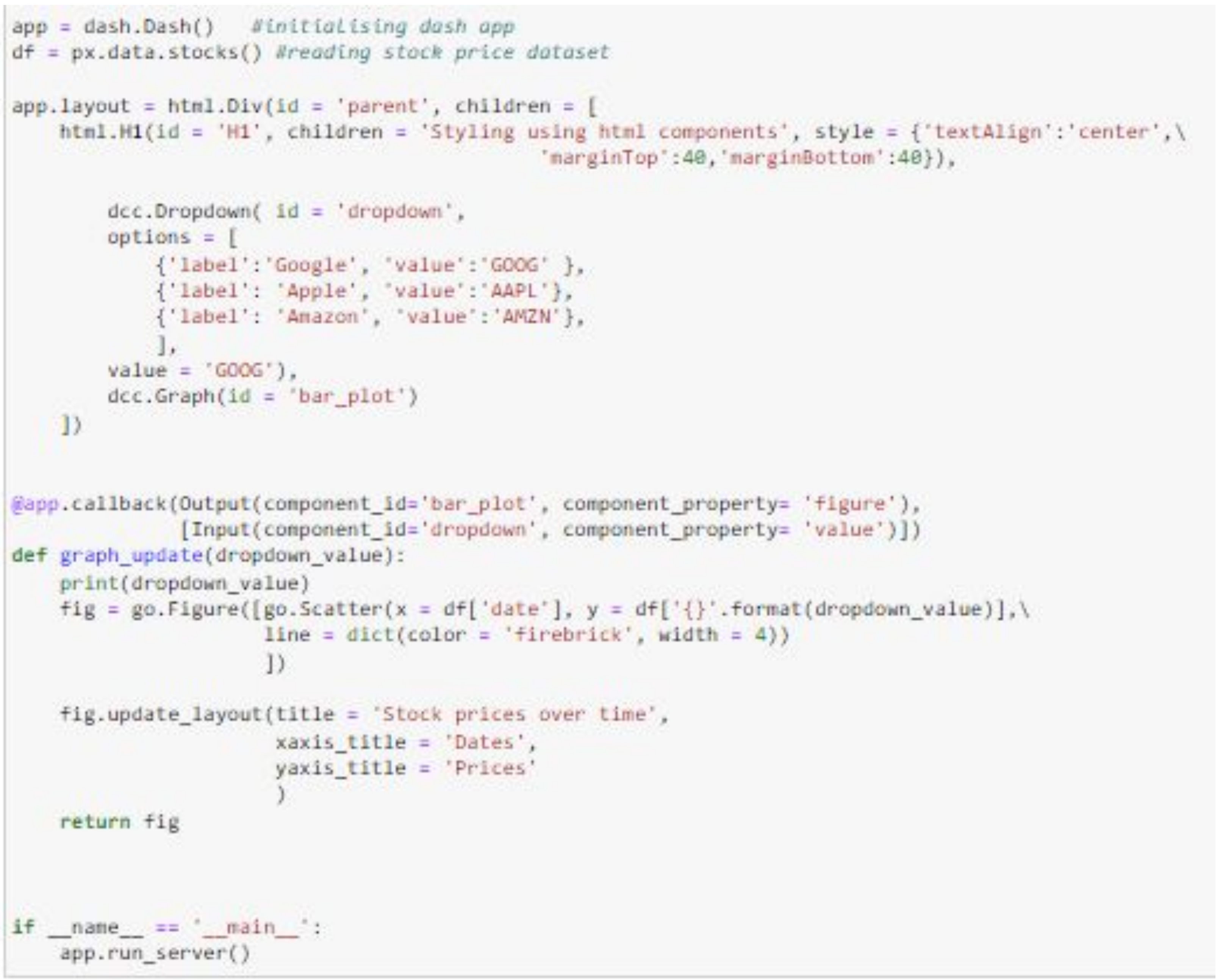
Dash Application

7. Making dashboard interactive using call back:

```
from dash.dependencies import Input, Output

@app.callback(Output(component_id='line_plot', component_property= 'figure'),
          [Input(component_id='dropdown', component_property= 'value')])
def graph_update(dropdown_value):
   print(dropdown_value)
   fig = go.Figure([go.Scatter(x = df['date'], y = df['{}'.format(dropdown_value)],\
               line = dict(color = 'firebrick', width = 4))
               ])

   fig.update_layout(title = 'Stock prices over time',
                xaxis_title = 'Dates',
                yaxis_title = 'Prices'
                )
   return fig
```

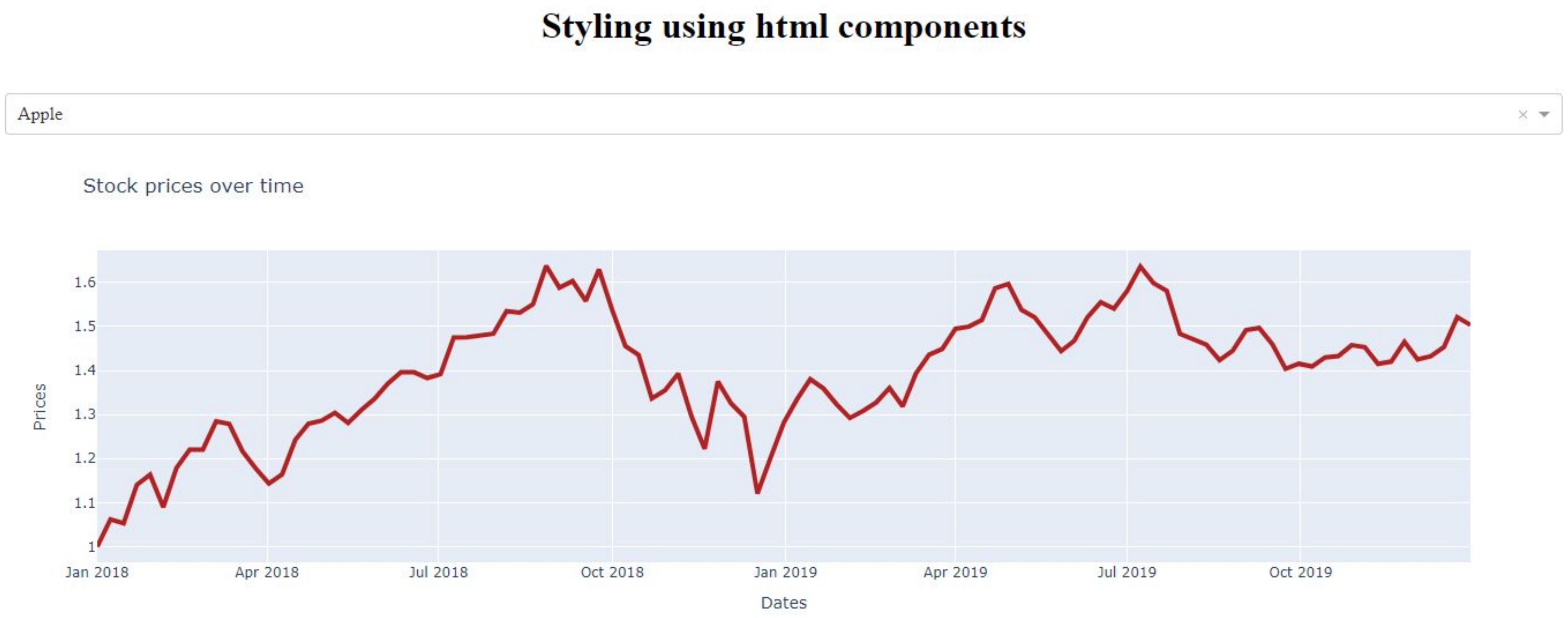
Dash Application

8. Combining Layout, Dropdown and Callback:



```python
app = dash.Dash()   #initialising dash app
df = px.data.stocks() #reading stock price dataset

app.layout = html.Div(id = 'parent', children = [
    html.H1(id = 'H1', children = 'Styling using html components', style = {'textAlign':'center',\
                                            'marginTop':40,'marginBottom':40}),

        dcc.Dropdown( id = 'dropdown',
        options = [
            {'label':'Google', 'value':'GOOG' },
            {'label': 'Apple', 'value':'AAPL'},
            {'label': 'Amazon', 'value':'AMZN'},
            ],
        value = 'GOOG'),
        dcc.Graph(id = 'bar_plot')
    ])


@app.callback(Output(component_id='bar_plot', component_property= 'figure'),
              [Input(component_id='dropdown', component_property= 'value')])
def graph_update(dropdown_value):
    print(dropdown_value)
    fig = go.Figure([go.Scatter(x = df['date'], y = df['{}'.format(dropdown_value)],\
                     line = dict(color = 'firebrick', width = 4))
                     ])

    fig.update_layout(title = 'Stock prices over time',
                      xaxis_title = 'Dates',
                      yaxis_title = 'Prices'
                      )
    return fig



if __name__ == '__main__':
    app.run_server()
```

Dash Application


Styling using html components

Apple

Adding Two Figures in the Dashboard

```
import dash
from dash import dcc,html
from dash.dependencies import Input, Output
import pandas as pd
import plotly.express as px
app = dash.Dash(__name__)
df_bar = pd.DataFrame({
   "Season": ["Summer", "Winter", "Autumn", "Spring"],
   "Rating": [3,2,1,4]
})
fig = px.bar(df_bar, x="Season", y="Rating", barmode="group")
```

Adding Two Figures in the Dashboard

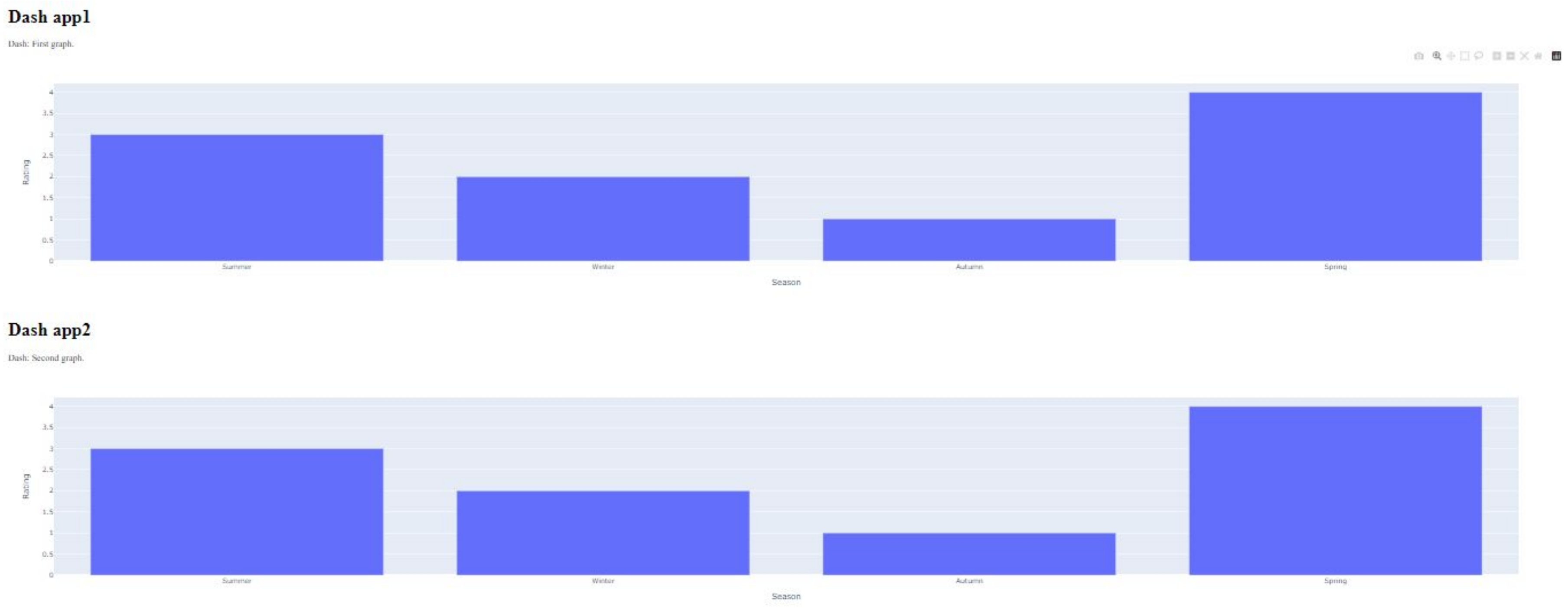
```
app.layout = html.Div(children=[
  # elements from the top of the page
  html.Div([
    html.H1(children='Dash app1'),
    html.Div(children='''
    Dash: First graph.'''),

    dcc.Graph(
      id='graph1',
      figure=fig
    ),
  ]),
```

Adding Two Figures in the Dashboard

```
# New Div for all elements in the new 'row' of the page
 html.Div([
   html.H1(children='Dash app2'),
   html.Div(children='''
   Dash: Second graph. '''),
dcc.Graph(
     id='graph2',
     figure=fig
   ),
 ]),
])
```

Adding Two Figures in the Dashboard

```
if __name__ == '__main__’:
    app.run(debug=True, use_reloader=False)
```


Dash app1

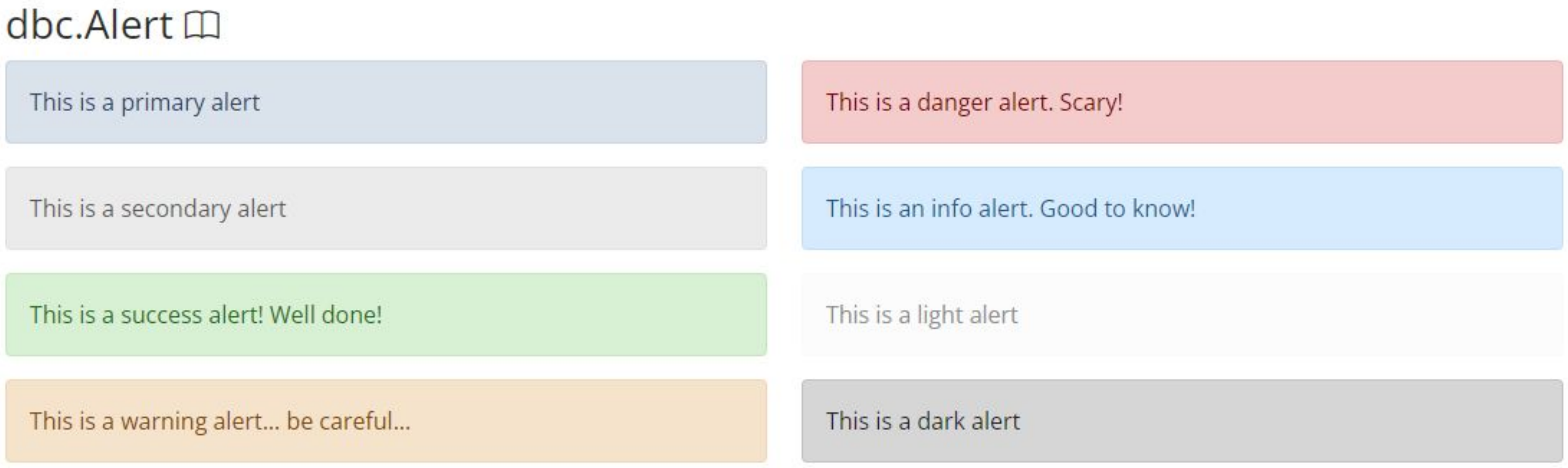
Dash: First graph.

Dash app2

Dash: Second graph.

Dash Bootstrap Theme Explorer

1. Components Gallery:


dbc.Alert

This is a primary alert

This is a danger alert. Scary!

This is a secondary alert

This is an info alert. Good to know!

This is a success alert! Well done!

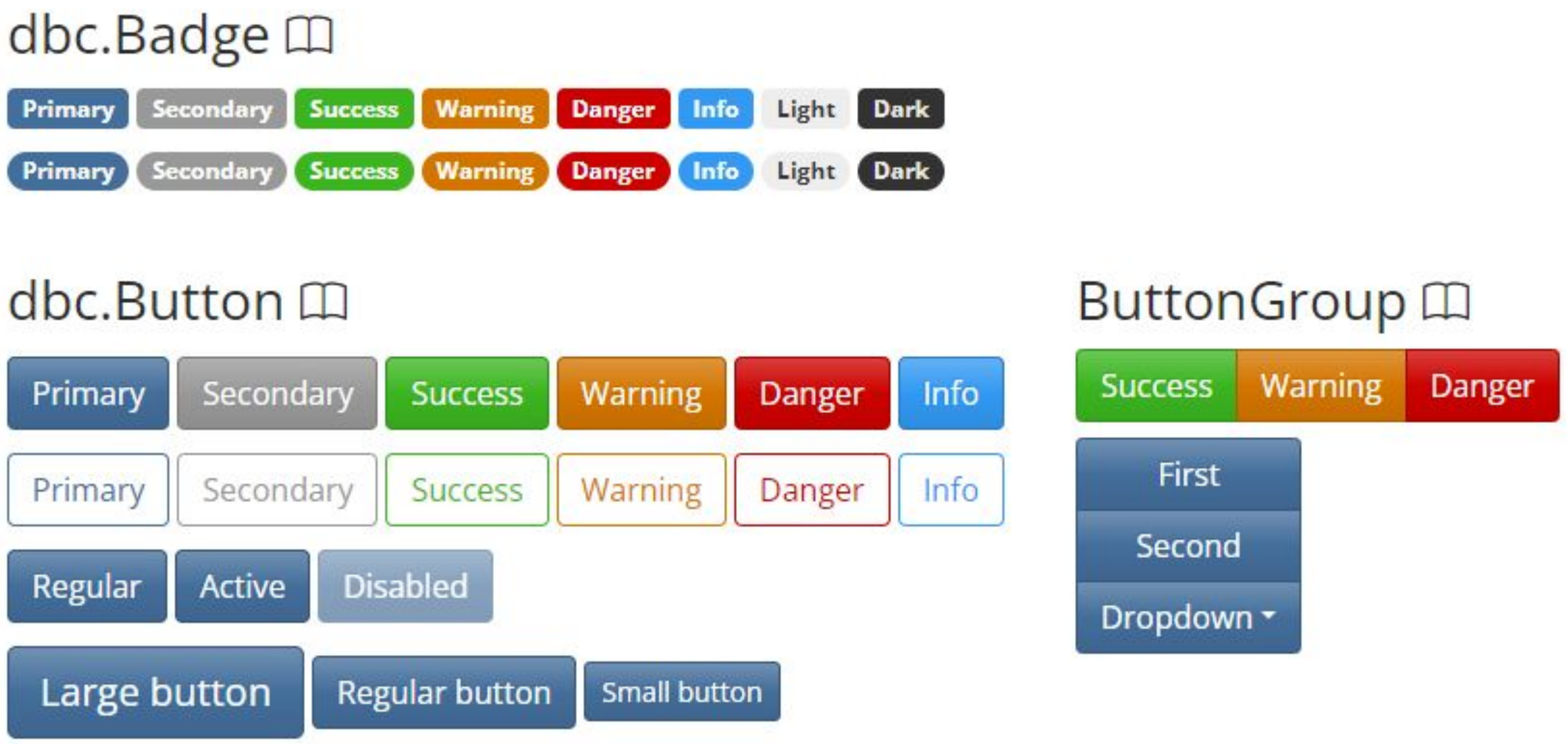
This is a light alert

This is a warning alert... be careful...

This is a dark alert

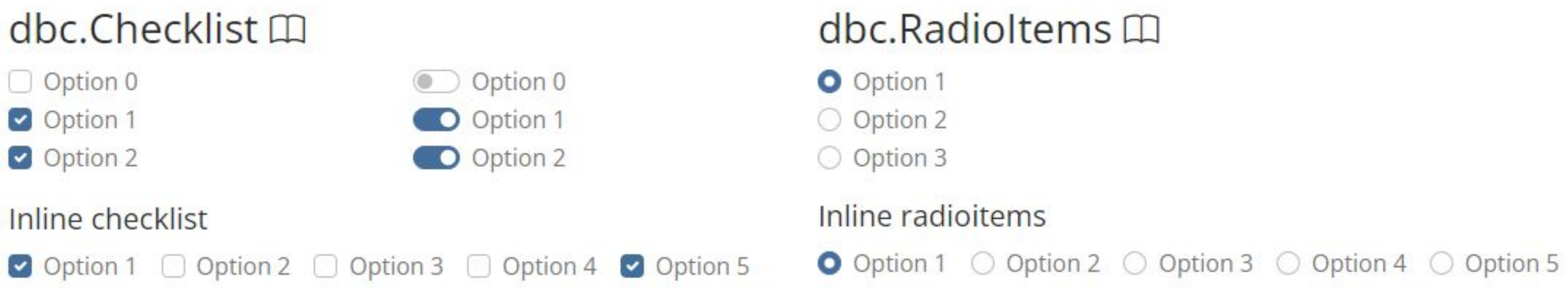
Dash Bootstrap Theme Explorer

1. Components Gallery:


dbc.Badge
Primary Secondary Success Warning Danger Info Light Dark
Primary Secondary Success Warning Danger Info Light Dark
dbc.Button
Primary Secondary Success Warning Danger Info
Primary Secondary Success Warning Danger Info
Regular Active Disabled
Large button Regular button Small button
ButtonGroup
Success Warning Danger
First
Second
Dropdown

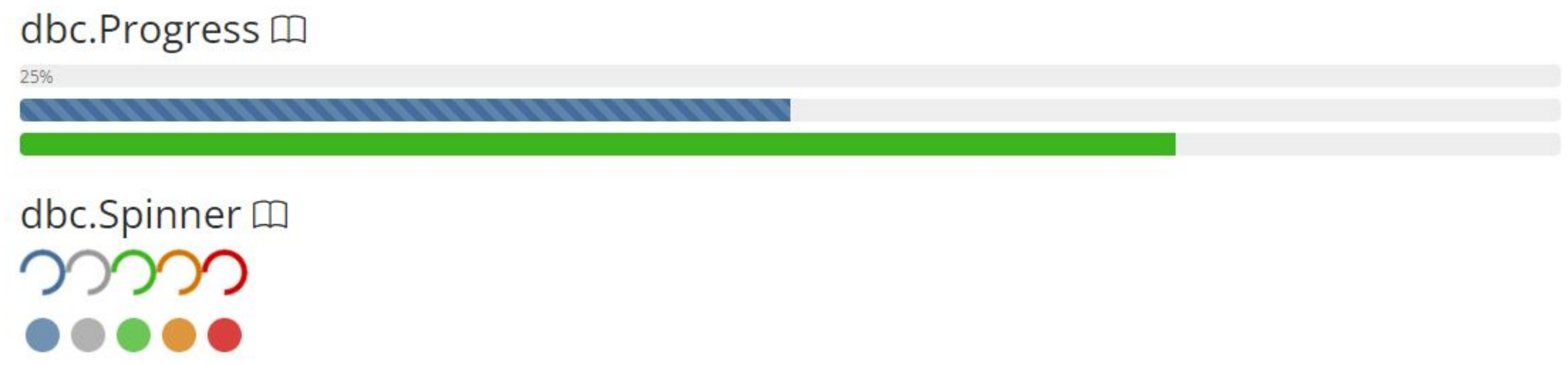
Dash Bootstrap Theme Explorer

1. Components Gallery:



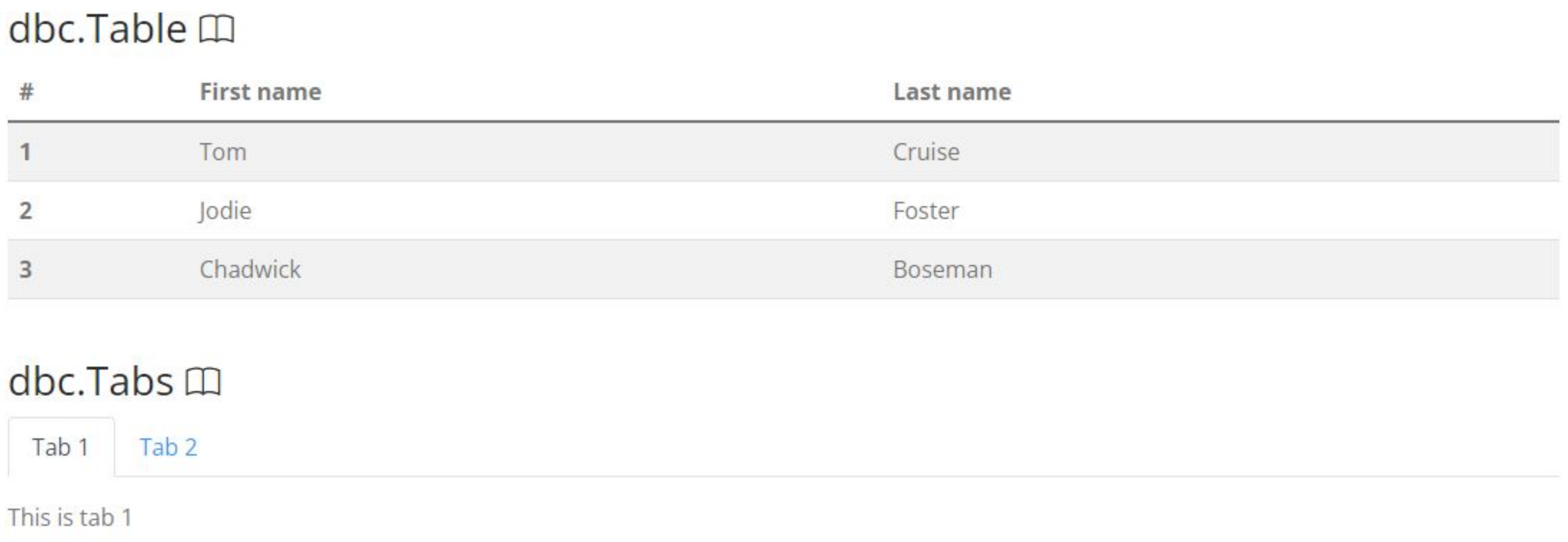
Dash Bootstrap Theme Explorer

1. Components Gallery:



Dash Bootstrap Theme Explorer

1. Components Gallery:



dbc.Table

#	First name	Last name
1	Tom	Cruise
2	Jodie	Foster
3	Chadwick	Boseman

dbc.Tabs

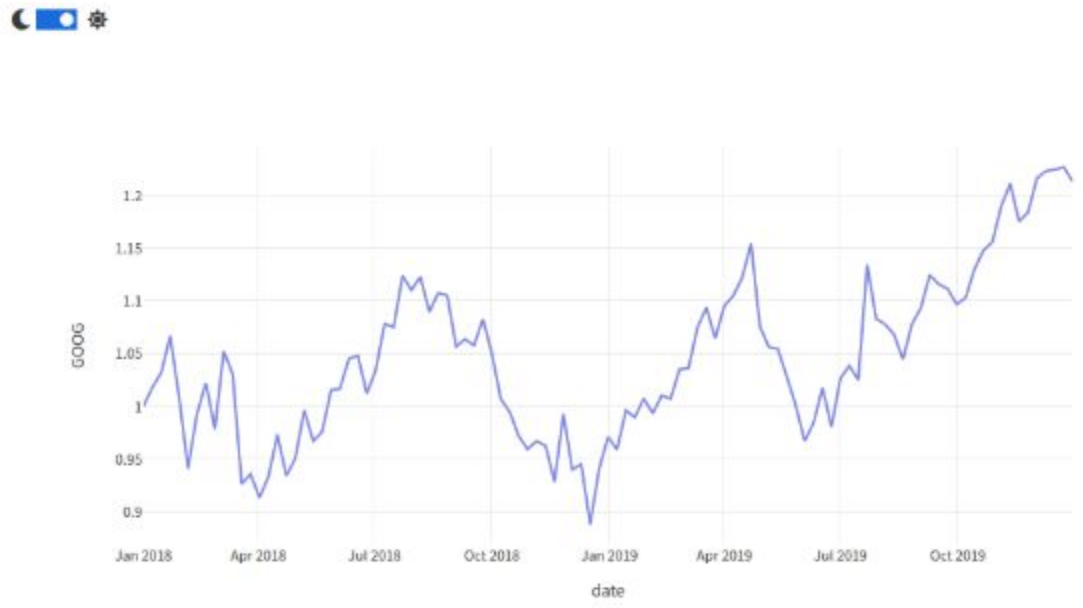
This is tab 1

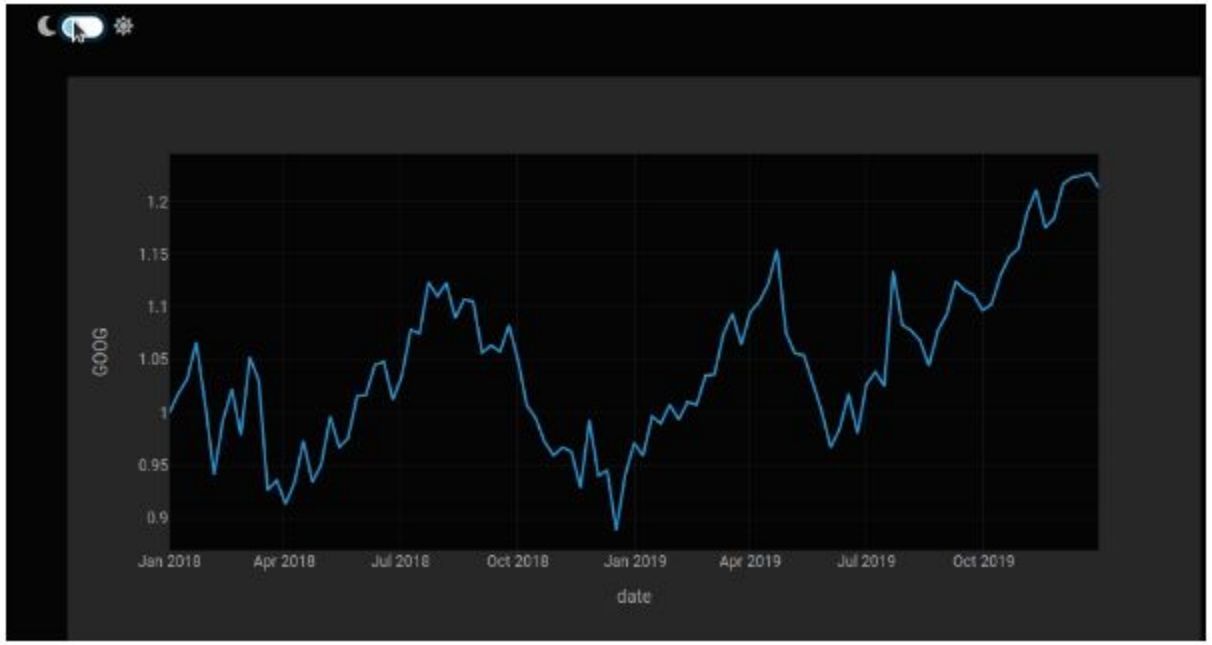
Dash Bootstrap Theme Explorer

2. Theme Change Components:
   - The dash-bootstrap-templates library has Two All-in-One components to change themes in a Dash app.
     - ThemeSwitchAIO toggles between two themes.
     - ThemeChangerAIO select from multiple themes.

Dash Bootstrap Theme Explorer

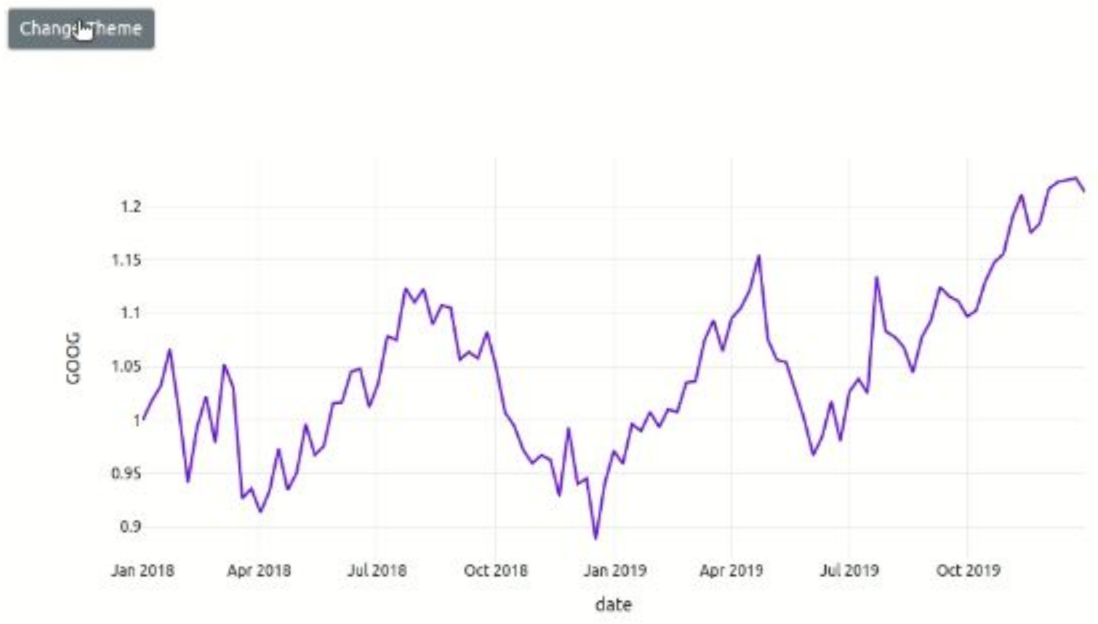
2. Theme Change Components:
   - ThemeSwitchAIO Example
     - ThemeSwitchAIO(aio_id="theme", themes=[dbc.themes.COSMO, dbc.themes.CYBORG])

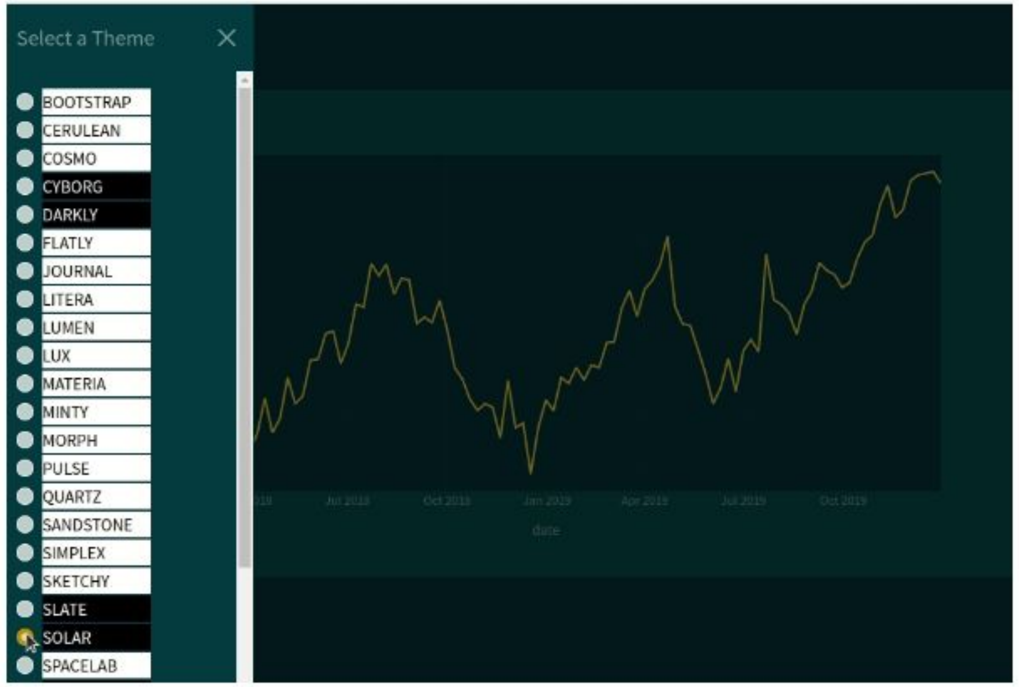




Dash Bootstrap Theme Explorer

2. Theme Change Components:
   - ThemeChangerAIO Example
     - `from dash_bootstrap_templates import ThemeChangerAIO, template_from_url`
     - `ThemeChangerAIO(aio_id="theme")`





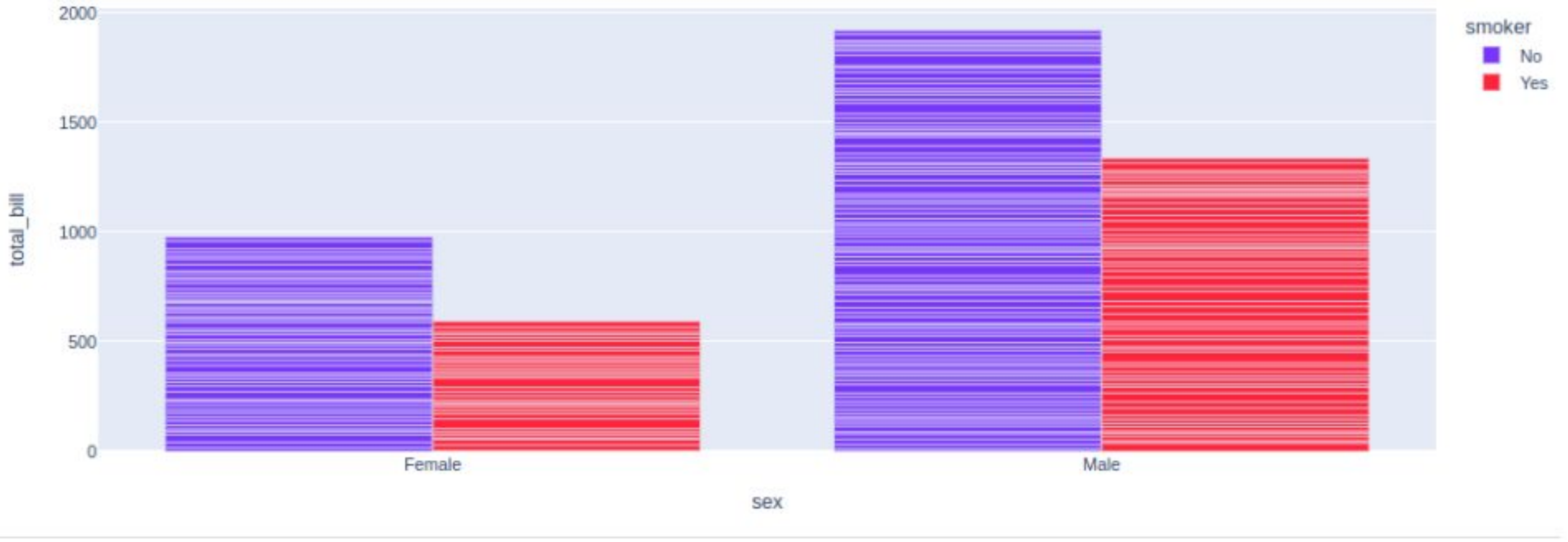
Dash Bootstrap Theme Explorer

3. Figure Templates:
   - Plotly Default Figure Template:

```
from dash import Dash, dcc, html
import plotly.express as px

df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group")

app=Dash(__name__)
app.layout = html.Div(dcc.Graph(figure=fig))

if __name__ == "__main__":
    app.run(debug=True)
```

Dash Bootstrap Theme Explorer

3. Figure Templates:
   - Plotly Default Figure Template:



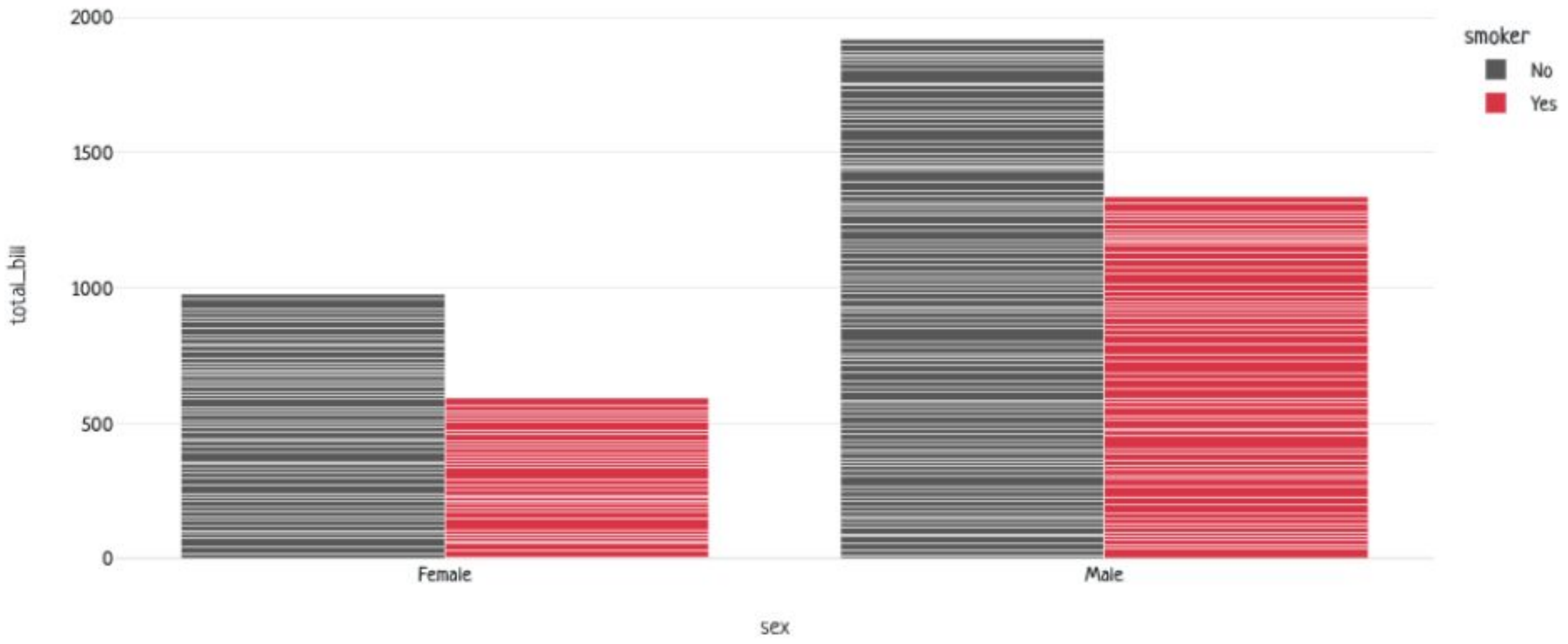
Dash Bootstrap Theme Explorer

3. Figure Templates:

```
from dash_bootstrap_templates import load_figure_template
load_figure_template("sketchy")
app=Dash(__name__, external_stylesheets=[dbc.themes.SKETCHY])
```

Dash Bootstrap Theme Explorer

3. Figure Templates:



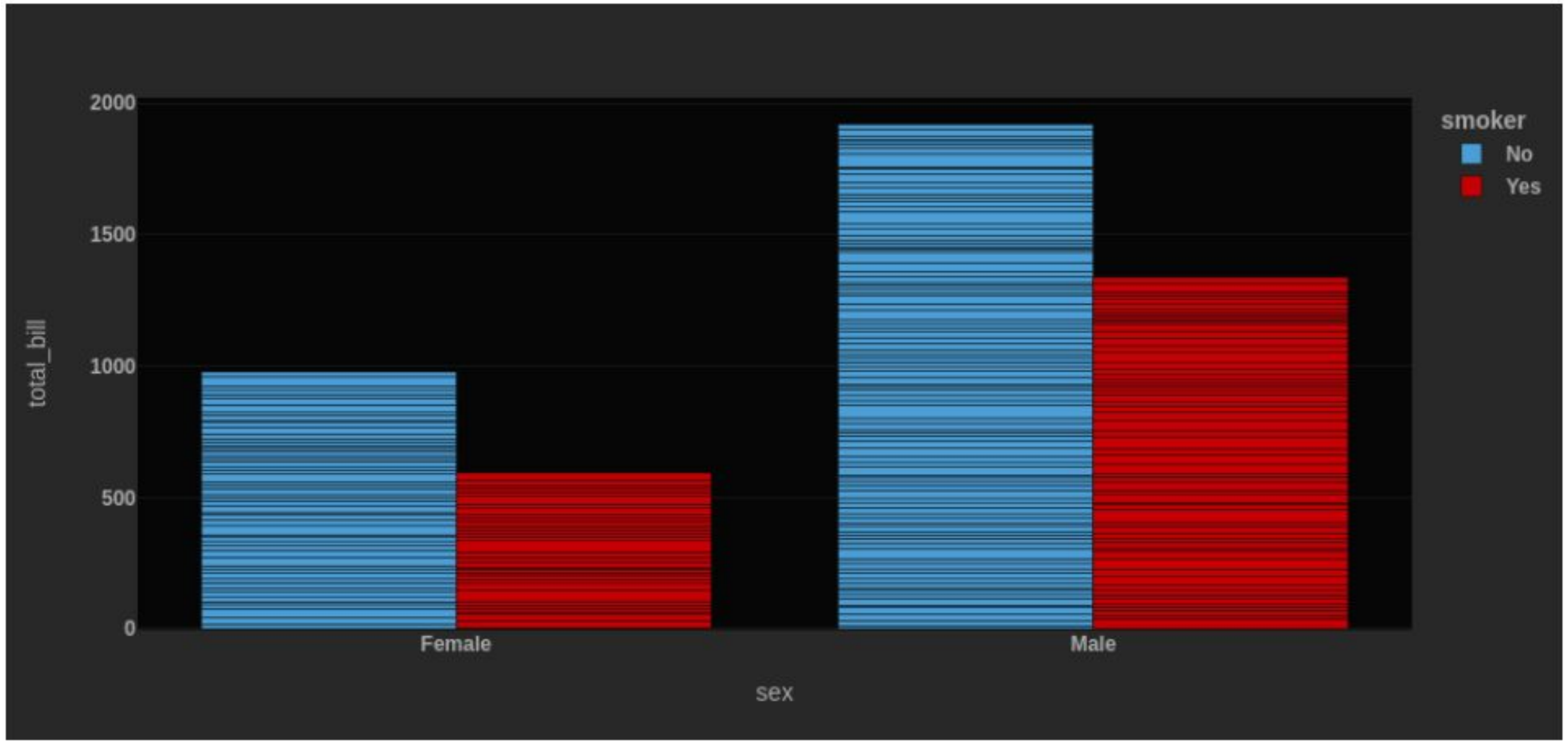
Dash Bootstrap Theme Explorer

3. Figure Templates:
    - Loading multiple figure templates:

```
    load_figure_template(["sketchy", "cyborg", "minty"])
    fig = px.bar(df, x="sex", y="total_bill", color="smoker",
barmode="group",    template="cyborg")
```

Dash Bootstrap Theme Explorer

3. Figure Templates:
   - Loading multiple figure templates:

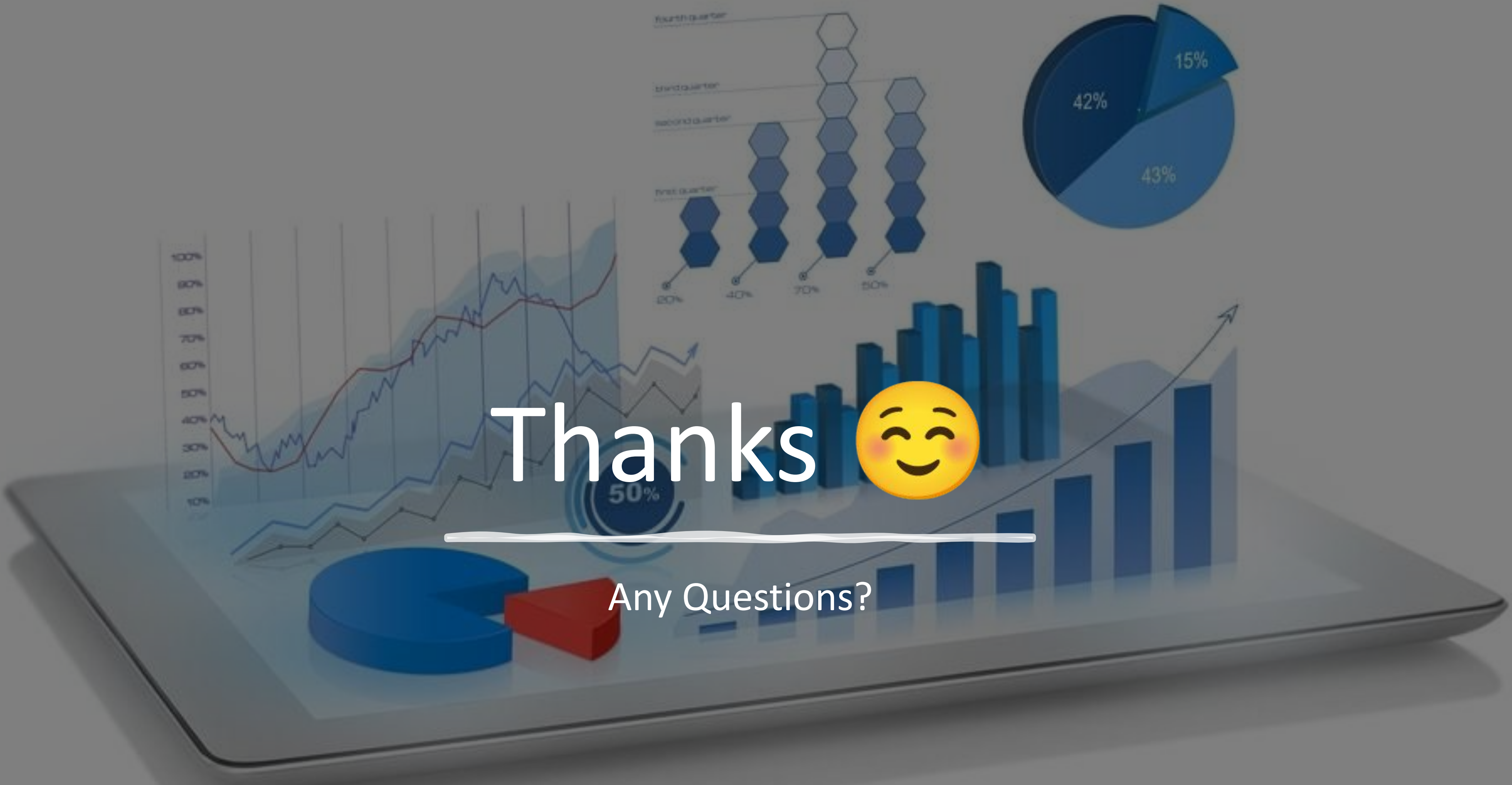


Dash Bootstrap Theme Explorer

4. Sample App:
    - Click on this link to check a dashboard template.
    - Check the attached file “SampleAppCode-Lecture 5.ipynb” for the code.

Dash Bootstrap Theme Explorer

- **Bonus task for one mark:**
  - **Description:** Implement the SampleApp by yourself to create a dashboard and show it to the TA during lecture 6.
  - **Due Date:** 11-Nov-2025.



Thanks ☺️

Any Questions?