

case_4.2

May 13, 2020

1 How are different types of crimes distributed by district across time in Boston?

1.1 Introduction (5 mts)

Business Context. You are a data consultant for the Boston Police Department. The department is looking to optimize its police deployment strategy so that it can tackle the most crimes as they occur with the fewest number of resources. Furthermore, different groups within the department specialize in different types of crimes, so this must play into their overall strategy. They would like to look at crime statistics by district and by month, as well as visualize them on a map by crime type and by date.

Business Problem. Your task is to **make an interactive dashboard that department heads can use to visualize crimes in the manners they indicated above.** The dashboard should be usable via a web browser like Chrome on the company's intranet.

Analytical Context. In the current case, we will again be using Dash by Plotly to develop the dashboard. This time, we will be using the Kaggle dataset titled "Crimes in Boston" found here: <https://www.kaggle.com/AnalyzeBoston/crimes-in-boston>. Unlike the first case, we will put this case together in an `app.py` file. We will also be using a SQLite database as the source of our data.

The case is structured as follows: you will (1) understand the data and plan the appropriate setup of the app to address the business problem; (2) set up the map component and various selectors to filter on the crime data; (3) set up a few additional plots to visualize crime by geography and time; and finally (4) use our completed app to derive insights.

1.2 Reading the data source (10 mts)

```
[9]: import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('sqlite:///crime.db')
df = pd.read_sql("SELECT * from crime LIMIT 100", engine.connect(),
                 parse_dates=['OCCURRED_ON_DATE'])
print(df.head())
```

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	\
0	I182070945	619	Larceny	LARCENY ALL OTHERS	

1	I182070943	1402	Vandalism	VANDALISM	
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	
3	I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	
4	I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	
DISTRICT REPORTING_AREA SHOOTING OCCURRED_ON_DATE YEAR MONTH \					
0	D14	808	None 2018-09-02 13:00:00	2018	9
1	C11	347	None 2018-08-21 00:00:00	2018	8
2	D4	151	None 2018-09-03 19:27:00	2018	9
3	D4	272	None 2018-09-03 21:16:00	2018	9
4	B3	421	None 2018-09-03 21:05:00	2018	9
DAY_OF_WEEK HOUR UCR_PART STREET Lat Long \					
0	Sunday	13	Part One LINCOLN ST	42.357791	-71.139371
1	Tuesday	0	Part Two HECLA ST	42.306821	-71.060300
2	Monday	19	Part Three CAZENOUE ST	42.346589	-71.072429
3	Monday	21	Part Three NEWCOMB ST	42.334182	-71.078664
4	Monday	21	Part Three DELHI ST	42.275365	-71.090361
Location					
0	(42.35779134, -71.13937053)				
1	(42.30682138, -71.06030035)				
2	(42.34658879, -71.07242943)				
3	(42.33418175, -71.07866441)				
4	(42.27536542, -71.09036101)				

We are going to use SQLAlchemy to read from the `sqlite` source. We have the following features:

1. **INCIDENT_NUMBER**: The ID of the incident
2. **OFFENSE_CODE**: Unique code for offense
3. **OFFENSE_CODE_GROUP**: Category of offense
4. **OFFENSE_DESCRIPTION**: Longer description of the offense
5. **DISTRICT**: Police district where crime was committed
6. **REPORTING_AREA**: Area where crime was reported
7. **SHOOTING**: Whether guns were fired during the incident
8. **OCCURED_ON_DATE**: Date of incident
9. **YEAR**: Year of incident
10. **MONTH**: Month of incident
11. **DAY_OF_WEEK**: Day of the week of incident
12. **HOUR**: Hour of incident
13. **UCR_PART**: Crime part code
14. **STREET**: Street of incident
15. **Lat**: Location (latitude) of incident
16. **Long**: Location(longitude) of incident
17. **Location**: Tuple of Lat and Long

1.2.1 Exercise 1: (5 mts)

Looking again at the business question and context, which of the above would be important to us?

Answer. The department is interested in crime patterns by district and by month (and not more granular than that). This means that DISTRICT and MONTH are important to us, but DAY_OF_WEEK, HOUR, REPORTING_AREA, and STREET are not (because they are too granular). However, YEAR is relevant because it is a higher-level clarification of MONTH (e.g. "December 2015" is an important clarification of "December"). OFFENSE_CODE_GROUP is important because the department has different groups that specialize in different types of crimes, so analyzing that ensures that we don't deploy the wrong specialists to the wrong locations.

Lat and Long are important because in order to put together a map of incidents, we need the latitude and longitude. OCCURRED_ON_DATE is important because the map needs to have a date filter.

INCIDENT_NUMBER and OFFENSE_CODE are identifiers and do not matter for the aggregate-level analysis the department cares about. OFFENSE_DESCRIPTION is a more granular version of OFFENSE_CODE_GROUP and likely does not matter. SHOOTING and UCR_PART are not even hinted at as being important.

1.3 Setting up our app (15 mts)

Create a new file called `app.py`. Let's start by importing the libraries required by the dashboard. We are going to do the core imports from Dash and add imports from `pandas` and `SQLAlchemy` to manage our data sources. Copy the following lines into the head of your `app.py` file:

```
[10]: from sqlalchemy import create_engine
import pandas as pd
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.graph_objects as go
```

1.3.1 Exercise 2: (5 mts)

Write code in your `app.py` file which will instantiate our Dash app, set up the token for the Mapbox plot, and instantiate the DataFrame which holds the `crime.db` data used by our app layout selectors. (Make sure the `app.py` file is in the same folder as the `crime.db` file!)

Answer. Our recommended solution is given below:

```
token = 'pk.eyJ1IjoibmV3dXNlcmZvcmV2ZXIiLCJhIjoiY2o2M3d1dTZiMGZobzMzbnp2Z2NiN3lmdyJ9.cQFKe3F3o
app = dash.Dash(__name__, external_stylesheets=['https://codepen.io/uditagarwal/pen/oNvwKNP.css'])

engine = create_engine('sqlite:///crime.db')
df = pd.read_sql("SELECT * from crime", engine.connect(), parse_dates=('OCCURRED_ON_DATE',))
```

1.3.2 Setting up the layout (5 mts)

The next step is to set the layout object. We are going to be using the same CSS and layout template as in the previous case, which is composed of a header and a body. The header contains the title of the dashboard and the body will contain the selectors, plots, and texts for our dashboard.

We will begin by setting up two divs. The first one will encase the header titled "Boston Crime Analysis", and the other div will represent the body, which will have no elements inside it for now. Copy the following into your `app.py` file:

```
app.layout = html.Div(children=[  
    html.Div(  
        children=[  
            html.H2(children="Boston Crime Analysis", className='h2-title'),  
        ],  
        className='study-browser-banner row'  
    ),  
    html.Div(  
        className="row app-body",  
        children=[]  
    )  
])
```

Our app is beginning to take shape. If you run this file and navigate to the Dash app in your browser, you will see a header and an empty body.

1.3.3 Planning our dashboard (5 mts)

Before we proceed, we should decide how to section the body of our app to best address business needs.

1. The department would like to be able to visualize crime on a map by crime type and by date. Thus, it makes sense to have some sort of drop-down filter by crime type associated with the map. And similar to the previous case, having a date range selector would allow users to filter by date.
2. The department has also stated that they would like to view crimes by district and by month. Since month is a time series variable, it makes sense to show this on some sort of graph which naturally shows the progression of time – a line plot comes to mind. Since district is a categorical variable, it makes sense to use a graph type that categorizes easily – a bar chart comes to mind.

Together, these imply the following sectioning of our app:

1. Section 1
 - Left card - Date range selector and crime type drop-down selector
 - Right card - Map of Boston
2. Section 2
 - Left card - Line plot of incidents in Boston by month
 - Right card - Bar chart of incidents by district

1.4 Setting up the first section of our dashboard (15 mts)

First, let's add the left card for the first section of our app, which will contain the selectors. These are the filters for our app, and determine the subset of our data which will be shown on our map and plots.

These components are going to be added to a div under the body div added before. Let's first add the date range selector to the layout:

```
app.layout = html.Div(children=[  
    html.Div(  
        children=[  
            html.H2(children="Boston Crime Analysis", className='h2-title'),  
        ],  
        className='study-browser-banner row'  
    ),  
    html.Div(  
        className="row app-body",  
        children=[  
            html.Div(  
                className="twelve columns card",  
                children=[  
                    html.Div(  
                        className="four columns card",  
                        children=[  
                            html.Div(  
                                className="bg-white user-control",  
                                children=[  
                                    html.Div(  
                                        className="padding-top-bot",  
                                        children=[  
                                            html.H6("Select a Date"),  
                                            dcc.DatePickerRange(  
                                                id="date-range",  
                                                start_date=df['OCCURRED_ON_DATE'].min(),  
                                                end_date=df['OCCURRED_ON_DATE'].max()  
                                            ),  
                                        ],  
                                    ),  
                                ],  
                            ),  
                        ],  
                    ),  
                ],  
            ),  
            html.Div(className="eight columns card", children=[])  
        ]  
    )  
)
```

])

Note that we added three divs to the body. The first is called `twelve columns card` (because it spans twelve columns' worth of width). The children of this div are going to be two more divs, `four columns card` and `eight columns card`, respectively.

The first div (`four columns card`) contains the `DatePickerRange` selector with `start_date` and `end_date` set to the minimum and maximum possible values for an incident date (`OCCURED_ON_DATE`) from our dataset.

Replace the layout in your `app.py` file with the above and run it to see the updated layout.

1.4.1 Exercise 3: (10 mts)

Add a `Dropdown` selector to the `four columns card` div with id `study-dropdown` right above the date range selector. The selector should contain all the unique offense codes listed in the `OFFENSE_CODE_GROUP` column in our dataset. Set the default value of this offense to "Larceny". Use the Dash docs for reference at <https://dash.plot.ly/dash-core-components/dropdown>.

Answer. Our recommended solution is shown below:

```
app.layout = html.Div(children=[  
    html.Div(  
        children=[  
            html.H2(children="Boston Crime Analysis", className='h2-title'),  
        ],  
        className='study-browser-banner row'  
    ),  
    html.Div(  
        className="row app-body",  
        children=[  
            html.Div(  
                className="twelve columns card",  
                children=[  
                    html.Div(  
                        className="four columns card",  
                        children=[  
                            html.Div(  
                                className="bg-white user-control",  
                                children=[  
                                    html.Div(  
                                        className="padding-top-bot",  
                                        children=[  
                                            html.H6("Select Crime Type"),  
                                            dcc.Dropdown(  
                                                id="study-dropdown",  
                                                multi=True,  
                                                value='Larceny',),  
                                                options=[{'label': label.title(), 'value': labo
```

```
        ),
        html.H6("Select a Date"),
        dcc.DatePickerRange(
            id="date-range",
            start_date=df['OCCURRED_ON_DATE'].min(),
            end_date=df['OCCURRED_ON_DATE'].max()
        ),
        ],
        ),
        ],
        ),
        ],
        ),
        ],
        ),
        ],
        ),
        html.Div(className="eight columns card", children=[])
    ]
)
]
)
])
```

1.5 Adding the Boston crime map (30 mts)

Let's add the map to our dashboard using the Mapbox scatterplot (<https://plot.ly/python/scattermapbox/>). We will add the Map plot to the `eight columns card` div.

NOTE: Our dataset has 300,000+ rows. If we load all of these data points at once, it will blow up the map. To get around this problem, we will only be plotting the data points corresponding to the crime groups selected on our dashboard in the crime selector filter.

The Map currently doesn't have any data passed to it for display. We will be updating the points shown on the map using a callback function which takes inputs from the Dropdown and

DatePickerRange selectors. Let's add the skeleton of the callback function to our app:

```
[ ]: # NOTE: Do NOT run this cell!!! It is for instructional purposes only - it will
  ↪NOT work!
@app.callback(
    dash.dependencies.Output('map-plot', 'figure'), # component with id
  ↪map-plot will be changed, the 'figure' argument is updated
    [
        dash.dependencies.Input('date-range', 'start_date'), # input with id
  ↪date-picker-range and the start_date parameter
        dash.dependencies.Input('date-range', 'end_date'),
        dash.dependencies.Input('study-dropdown', 'value'),
    ]
)
def update_crimes_map(start_date, end_date, value):
    return {
        'data': locations_by_crimetype(value, start_date, end_date),
        'layout': go.Layout(
            mapbox_style="dark",
            mapbox_accesstoken=token,
            mapbox_zoom=10,
            margin={'t': 0, 'l': 0, 'r': 0, 'b': 30},
            mapbox_center={"lat": df['Lat'][0], "lon": df['Long'][0]}
        )
    }
```

Let's take a bit to talk about the code above. We created a function called `update_crimes_map()` with the `app.callback()` decorator. The inputs to the decorator are:

1. date range input as `start_date` value
2. date range input as `end-date` value
3. crime type dropdown with `value`

The output of the function updates the `map-plot`. The callback function will need to return the value for `figure` on the map plot. The value of `figure` should be the data (which is the plot) and the layout of this plot.

As you might have noticed, the `data` calls the function `locations_by_crimetype()` which doesn't exist yet.

Let's set up this function which is going to return a `go.Scattermapbox()` object containing information about the latitude and longitude the incidents. To do this, we will run a SQL Query to filter the rows of our data source based on the data range selector and crime type dropdown selector. Note that our multi-value crime type selector can pass more than one type of crime to the map.

1.5.1 Exercise 4: (15 mts)

Write the `get_filtered_rows()` function, which takes `crime_type`, `start_date`, and `end_date` as inputs and return a `pandas` DataFrame containing the rows of our data source which fall within that `crime_type` and between `start_date` and `end_date`. (Hint: you may need to look up the syntax for the `read_sql()` function for this.)

Answer. Our recommended solution is given below:

```
[9]: def get_filtered_rows(crime_type, start_date, end_date):
    crime_len = ','.join('?'*len(crime_type))
    sql_query = f'SELECT * from crime WHERE OCCURRED_ON_DATE BETWEEN ? and ?' + \
    'AND OFFENSE_CODE_GROUP in ({crime_len})'

    sql_params = [start_date, end_date]
    for each in crime_type:
        sql_params.append(each)

    return pd.read_sql(sql_query, engine.connect(), params=sql_params, \
    parse_dates=('OCCURRED_ON_DATE',))
```

Now we can use this function to write the `locations_by_crimeType()` function:

```
[10]: import random

def locations_by_crimeType(crime_type, start_date, end_date):
    data = [] # Output of the function is an array

    df = get_filtered_rows(crime_type, start_date, end_date)
    for name, group in df.groupby('OFFENSE_CODE_GROUP'):
        color = "%06x" % random.randint(0, 0xFFFFFF)
        data.append(
            go.Scattermapbox(
                lat=group['Lat'],
                lon=group['Long'],
                mode='markers',
                marker={
                    'color': '#' + color,
                },
                text=group['OFFENSE_DESCRIPTION'],
                name=name
            )
        )
    return data
```

Let's put this all together in our `app.py` file and run it.

1.5.2 Question:

Using the latest version of your `app.py` file, see if you can find out the region where:

1. The most license violations happened?
 2. The most aircraft offenses happened?
 3. The most liquor violations happened?

1.6 Setting up section 2 of our dashboard (15 mts)

We can now proceed to setting up the next section of our dashboard, which will provide crime statistics by district and by month. We will add another div called `twelve columns card 2` which contains two six-column wide divs:

1. The first div will be a line chart which tracks the total number of incidents per month over time
 2. The second div will be a bar chart of the number of incidents by district

Both will be filtered based on the crime type dropdown selector choice.

1.6.1 Exercise 4: (10 mts)

Add the following two plots to the `six columns` card and `six columns` card 2 divs:

1. First div - `go.Scatter()` with the id `crime-total-graph` (remember, `go.Scatter()` is used to generate a line plot in Dash!)
2. Second div - `go.Bar()` with the id `crime-district-graph`

For now, don't worry about passing any data into these graphs. Later, we will write callback functions based on our input selectors in order to update them.

Answer. Our recommended solution is shown below:

```
app.layout = html.Div(children=[  
    html.Div(  
        children=[  
            html.H2(children="Boston Crime Analysis", className='h2-title'),  
        ],  
        className='study-browser-banner row'  
    ),  
    html.Div(  
        className="row app-body",  
        children=[  
            html.Div(  
                className="twelve columns card",  
                children=[  
                    html.Div(  
                        className="four columns card",  
                        children=[  
                            html.Div(  
                                className="bg-white user-control",  
                                children=[  
                                    html.Div(  
                                        className="padding-top-bot",  
                                        children=[  
                                            html.H6("Select Crime Type"),  
                                            dcc.Dropdown(  
                                                id="study-dropdown",  
                                                multi=True,  
                                                value=('Larceny',),  
                                                options=[{'label': label.title(), 'value': labo  
                                            ),  
                                            html.H6("Select a Date"),  
                                            dcc.DatePickerRange(  
                                                id="date-range",  
                                                start_date=df['OCCURRED_ON_DATE'].min(),  
                                                end_date=df['OCCURRED_ON_DATE'].max()  
                                            ),  
                                        ],  
                                    ],  
                                ),  
                            ],  
                        ),  
                    ],  
                ),  
            ],  
        ),  
    ],  
)
```

```

        ),
        ],
        )
    ],
),
html.Div(className="eight columns card", children=[
    html.H1(children="Geographical Map of Crimes in Boston", style={'textA
        dcc.Graph(
            id='map-plot',
            figure={
                'data': [go.Scattermapbox()],
                'layout': go.Layout(
                    mapbox_style="dark",
                    mapbox_accesstoken=token,
                    mapbox_zoom=10,
                    margin={'t': 0, 'l': 0, 'r': 0, 'b': 30},
                    mapbox_center={"lat": df['Lat'][0], "lon": df['Long
                )
            }
        )
    ])
),
html.Div(
    className='twelve columns card 2',
    children=[
        html.Div(
            className='six columns card',
            children=[
                dcc.Graph(
                    id='crime-total-graph',
                    figure={
                        'data': [go.Scatter()],
                    }
                )
            ]
        ),
        html.Div(
            className='six columns card 2',
            children=[
                dcc.Graph(
                    id='crime-district-graph',
                    figure={
                        'data': [go.Bar()],
                    }
                )
            ]
        )
    ]
)
)

```

```

        ]
    )
]
])

```

1.7 Setting up the callback function for the crime by month graph (10 mts)

To update data in our line plot with id `crime-total-graph` (remember, a line plot in Dash is created using `go.Scatter()`), we will make a new callback function which takes the settings on our selectors as input.

1.7.1 Exercise 5: (5 mts)

Write the skeleton of such a callback function, which we'll call `update_crimes_total()`. Assume that there is already another function `crimes_by_year()` that you can use which returns the data necessary for your line plot.

Answer. Our recommended solution is shown below:

```

@app.callback(
    dash.dependencies.Output('crime-total-graph', 'figure'),
    [
        dash.dependencies.Input('date-range', 'start_date'), # input with id date-picker-range
        dash.dependencies.Input('date-range', 'end_date'),
        dash.dependencies.Input('study-dropdown', 'value'),
    ]
)
def update_crimes_total(start_date, end_date, value):
    return {
        'data': crimes_by_year(value, start_date, end_date),
        'layout': {
            'title': {
                'text': 'Crime Occurrence over Time'
            }
        }
    }

```

Of course, this won't work because `crimes_by_year()` hasn't been defined yet. Let's flesh it out so it can properly return the data our line plot needs:

```
[7]: def crimes_by_year(crime_type, start_date, end_date):
    data = []
    df = get_filtered_rows(crime_type, start_date, end_date)

    df['YearMonth'] = pd.to_datetime(df['OCCURRED_ON_DATE'].map(lambda x: x.strftime("%Y-%m").format(x.year, x.month)))
```

```

for name, group in df.groupby('OFFENSE_CODE_GROUP'):
    grouped = group.groupby('YearMonth', as_index=False).count()
    data.append(
        go.Scatter(x=grouped['YearMonth'], y=grouped['Lat'], name=name)
    )
return data

```

Go ahead and add this to your app.py file.

1.8 Setting up the callback function for the bar chart (15 mts)

Let's continue by setting up the callback for the bar chart as well. Now that you've seen it done once for the line chart, we'll leave this one to you.

1.8.1 Exercise 6: (15 mts)

Create a callback function for the bar chart with id `crime-district-graph`. This needs to display the number of incidents per district, filtered by the values in the crime type selector from section 1.

We want this to be a horizontal bar chart, with the x-axis representing the number of incidents and the y-axis representing the names of the districts. Refer to <https://plot.ly/python/bar-charts/> for the parameters of a bar chart.

Answer. Our recommended solution is shown below:

```

def crimes_by_district(crime_type, start_date, end_date):
    data = []
    df = get_filtered_rows(crime_type, start_date, end_date)

    for name, group in df.groupby('OFFENSE_CODE_GROUP'):
        grouped = group.groupby('DISTRICT', as_index=False).count()
        data.append(
            go.Bar(y=grouped['DISTRICT'], x=grouped['Lat'].sort_values(), name=name, orientation='h')
        )
    return data

@app.callback(
    dash.dependencies.Output('crime-district-graph', 'figure'),
    [
        dash.dependencies.Input('date-range', 'start_date'), # input with id date-picker-range
        dash.dependencies.Input('date-range', 'end_date'),
        dash.dependencies.Input('study-dropdown', 'value'),
    ]
)
def update_crimes_district_plot(start_date, end_date, value):
    return {

```

```

'data': crimes_by_district(value, start_date, end_date),
'layout': {
    'title': {
        'text': 'Crime Occurrence by District',
    }
}
}

```

1.8.2 Question:

Using the latest version of your `app.py` file, see if you can answer the following questions:

1. In which months are there the least number of crimes? Is this pattern consistent across various crime types?
2. What is the trend over time in "Residential Burglaries" crimes?
3. Which district has the highest number of Larceny incidents?

1.9 Adding hourly plots for each day of the week (15 mts)

You present your app to the department heads, who are very pleased with the results. However, they would like you to add one more thing: they want to see, for each day of the week, a line plot representing the total number of incidents per hour during that day of the week over the time period selected. That is, they want to see *subplots* (<https://plot.ly/python/subplots/>) for each day of the week.

Let's add a new div to our layout with an empty plot called `crimes-weekly`:

```

app.layout = html.Div(children=[
    html.Div(
        children=[
            html.H2(children="Boston Crime Analysis", className='h2-title'),
        ],
        className='study-browser-banner row'
    ),
    html.Div(
        className="row app-body",
        children=[
            html.Div(
                className="twelve columns card",
                children=[
                    html.Div(
                        className="four columns card",
                        children=[
                            html.Div(
                                className="bg-white user-control",
                                children=[
                                    html.Div(

```

```

        className="padding-top-bot",
        children=[
            html.H6("Select Crime Type"),
            dcc.Dropdown(
                id="study-dropdown",
                multi=True,
                value=('Larceny',),
                options=[{'label': label.title(), 'value': label}
                ],
                html.H6("Select a Date"),
                dcc.DatePickerRange(
                    id="date-range",
                    start_date=df['OCCURRED_ON_DATE'].min(),
                    end_date=df['OCCURRED_ON_DATE'].max()
                ),
            ],
        ],
    ),
    html.Div(className="eight columns card", children=[
        html.H1(children="Geographical Map of Crimes in Boston", style={'textA
        dcc.Graph(
            id='map-plot',
            figure={
                'data': [go.Scattermapbox()],
                'layout': go.Layout(
                    mapbox_style="dark",
                    mapbox_accesstoken=token,
                    mapbox_zoom=10,
                    margin={'t': 0, 'l': 0, 'r': 0, 'b': 30},
                    mapbox_center={"lat": df['Lat'][0], "lon": df['Long
                )
            }
        )
    ],
),
html.Div(
    className='twelve columns card 2',
    children=[
        html.Div(
            className='six columns card',
            children=[
                dcc.Graph(
                    id='crime-total-graph',
                    figure={

```

```
        'data': [go.Scatter()],
    }
)
]
),
html.Div(
    className='six columns card 2',
    children=[
        dcc.Graph(
            id='crime-district-graph',
            figure={
                'data': [go.Bar()],
            }
        )
    ]
)
],
html.Div(
    className='twelve columns card 3',
    children=[
        dcc.Graph(
            id="crimes-weekly",
            figure={
                'data': [go.Scatter()]
            }
        )
    ]
)
)
])
])
```

1.9.1 Adding the callback function for our line plots

We will setup a simple callback function which updates the `crimes-weekly` plot:

```
[4]: # NOTE: Do NOT run this cell!!! It is for instructional purposes only - it will  
      ↵NOT work!  
      @app.callback(  
          dash.dependencies.Output('crimes-weekly', 'figure'),  
          [  
              dash.dependencies.Input('date-range', 'start_date'), # input with id  
              ↵date-picker-range and the start_date parameter  
              dash.dependencies.Input('date-range', 'end_date'),  
              dash.dependencies.Input('study-dropdown', 'value'),  
          ]
```

```
)
def update_crimes_scatter_plot(start_date, end_date, value):
    return crimes_week(value, start_date, end_date)
```

↳

↳-----

```
NameError Traceback (most recent call last)

<ipython-input-4-191479f6adcf> in <module>()
----> 1 @app.callback(
      2     dash.dependencies.Output('crimes-weekly', 'figure'),
      3     [
      4         dash.dependencies.Input('date-range', 'start_date'), # input
      5         dash.dependencies.Input('date-range', 'end_date'),
      6     ],
      7     prevent_initial_call=True
      8 )
      9     return update_crimes_scatter_plot(start_date, end_date)
     10
     11
     12
     13
     14
     15
     16
     17
     18
     19
     20
     21
     22
     23
     24
     25
     26
     27
     28
     29
     30
     31
     32
     33
     34
     35
     36
     37
     38
     39
     40
     41
     42
     43
     44
     45
     46
     47
     48
     49
     50
     51
     52
     53
     54
     55
     56
     57
     58
     59
     60
     61
     62
     63
     64
     65
     66
     67
     68
     69
     70
     71
     72
     73
     74
     75
     76
     77
     78
     79
     80
     81
     82
     83
     84
     85
     86
     87
     88
     89
     90
     91
     92
     93
     94
     95
     96
     97
     98
     99
     100
     101
     102
     103
     104
     105
     106
     107
     108
     109
     110
     111
     112
     113
     114
     115
     116
     117
     118
     119
     120
     121
     122
     123
     124
     125
     126
     127
     128
     129
     130
     131
     132
     133
     134
     135
     136
     137
     138
     139
     140
     141
     142
     143
     144
     145
     146
     147
     148
     149
     150
     151
     152
     153
     154
     155
     156
     157
     158
     159
     160
     161
     162
     163
     164
     165
     166
     167
     168
     169
     170
     171
     172
     173
     174
     175
     176
     177
     178
     179
     180
     181
     182
     183
     184
     185
     186
     187
     188
     189
     190
     191
     192
     193
     194
     195
     196
     197
     198
     199
     200
     201
     202
     203
     204
     205
     206
     207
     208
     209
     210
     211
     212
     213
     214
     215
     216
     217
     218
     219
     220
     221
     222
     223
     224
     225
     226
     227
     228
     229
     230
     231
     232
     233
     234
     235
     236
     237
     238
     239
     240
     241
     242
     243
     244
     245
     246
     247
     248
     249
     250
     251
     252
     253
     254
     255
     256
     257
     258
     259
     260
     261
     262
     263
     264
     265
     266
     267
     268
     269
     270
     271
     272
     273
     274
     275
     276
     277
     278
     279
     280
     281
     282
     283
     284
     285
     286
     287
     288
     289
     290
     291
     292
     293
     294
     295
     296
     297
     298
     299
     300
     301
     302
     303
     304
     305
     306
     307
     308
     309
     310
     311
     312
     313
     314
     315
     316
     317
     318
     319
     320
     321
     322
     323
     324
     325
     326
     327
     328
     329
     330
     331
     332
     333
     334
     335
     336
     337
     338
     339
     340
     341
     342
     343
     344
     345
     346
     347
     348
     349
     350
     351
     352
     353
     354
     355
     356
     357
     358
     359
     360
     361
     362
     363
     364
     365
     366
     367
     368
     369
     370
     371
     372
     373
     374
     375
     376
     377
     378
     379
     380
     381
     382
     383
     384
     385
     386
     387
     388
     389
     390
     391
     392
     393
     394
     395
     396
     397
     398
     399
     400
     401
     402
     403
     404
     405
     406
     407
     408
     409
     410
     411
     412
     413
     414
     415
     416
     417
     418
     419
     420
     421
     422
     423
     424
     425
     426
     427
     428
     429
     430
     431
     432
     433
     434
     435
     436
     437
     438
     439
     440
     441
     442
     443
     444
     445
     446
     447
     448
     449
     450
     451
     452
     453
     454
     455
     456
     457
     458
     459
     460
     461
     462
     463
     464
     465
     466
     467
     468
     469
     470
     471
     472
     473
     474
     475
     476
     477
     478
     479
     480
     481
     482
     483
     484
     485
     486
     487
     488
     489
     490
     491
     492
     493
     494
     495
     496
     497
     498
     499
     500
     501
     502
     503
     504
     505
     506
     507
     508
     509
     510
     511
     512
     513
     514
     515
     516
     517
     518
     519
     520
     521
     522
     523
     524
     525
     526
     527
     528
     529
     530
     531
     532
     533
     534
     535
     536
     537
     538
     539
     540
     541
     542
     543
     544
     545
     546
     547
     548
     549
     550
     551
     552
     553
     554
     555
     556
     557
     558
     559
     560
     561
     562
     563
     564
     565
     566
     567
     568
     569
     570
     571
     572
     573
     574
     575
     576
     577
     578
     579
     580
     581
     582
     583
     584
     585
     586
     587
     588
     589
     590
     591
     592
     593
     594
     595
     596
     597
     598
     599
     600
     601
     602
     603
     604
     605
     606
     607
     608
     609
     610
     611
     612
     613
     614
     615
     616
     617
     618
     619
     620
     621
     622
     623
     624
     625
     626
     627
     628
     629
     630
     631
     632
     633
     634
     635
     636
     637
     638
     639
     640
     641
     642
     643
     644
     645
     646
     647
     648
     649
     650
     651
     652
     653
     654
     655
     656
     657
     658
     659
     660
     661
     662
     663
     664
     665
     666
     667
     668
     669
     670
     671
     672
     673
     674
     675
     676
     677
     678
     679
     680
     681
     682
     683
     684
     685
     686
     687
     688
     689
     690
     691
     692
     693
     694
     695
     696
     697
     698
     699
     700
     701
     702
     703
     704
     705
     706
     707
     708
     709
     710
     711
     712
     713
     714
     715
     716
     717
     718
     719
     720
     721
     722
     723
     724
     725
     726
     727
     728
     729
     730
     731
     732
     733
     734
     735
     736
     737
     738
     739
     740
     741
     742
     743
     744
     745
     746
     747
     748
     749
     750
     751
     752
     753
     754
     755
     756
     757
     758
     759
     760
     761
     762
     763
     764
     765
     766
     767
     768
     769
     770
     771
     772
     773
     774
     775
     776
     777
     778
     779
     780
     781
     782
     783
     784
     785
     786
     787
     788
     789
     790
     791
     792
     793
     794
     795
     796
     797
     798
     799
     800
     801
     802
     803
     804
     805
     806
     807
     808
     809
     810
     811
     812
     813
     814
     815
     816
     817
     818
     819
     820
     821
     822
     823
     824
     825
     826
     827
     828
     829
     830
     831
     832
     833
     834
     835
     836
     837
     838
     839
     840
     841
     842
     843
     844
     845
     846
     847
     848
     849
     850
     851
     852
     853
     854
     855
     856
     857
     858
     859
     860
     861
     862
     863
     864
     865
     866
     867
     868
     869
     870
     871
     872
     873
     874
     875
     876
     877
     878
     879
     880
     881
     882
     883
     884
     885
     886
     887
     888
     889
     890
     891
     892
     893
     894
     895
     896
     897
     898
     899
     900
     901
     902
     903
     904
     905
     906
     907
     908
     909
     910
     911
     912
     913
     914
     915
     916
     917
     918
     919
     920
     921
     922
     923
     924
     925
     926
     927
     928
     929
     930
     931
     932
     933
     934
     935
     936
     937
     938
     939
     940
     941
     942
     943
     944
     945
     946
     947
     948
     949
     950
     951
     952
     953
     954
     955
     956
     957
     958
     959
     960
     961
     962
     963
     964
     965
     966
     967
     968
     969
     970
     971
     972
     973
     974
     975
     976
     977
     978
     979
     980
     981
     982
     983
     984
     985
     986
     987
     988
     989
     990
     991
     992
     993
     994
     995
     996
     997
     998
     999
     1000
     1001
     1002
     1003
     1004
     1005
     1006
     1007
     1008
     1009
     1010
     1011
     1012
     1013
     1014
     1015
     1016
     1017
     1018
     1019
     1020
     1021
     1022
     1023
     1024
     1025
     1026
     1027
     1028
     1029
     1030
     1031
     1032
     1033
     1034
     1035
     1036
     1037
     1038
     1039
     1040
     1041
     1042
     1043
     1044
     1045
     1046
     1047
     1048
     1049
     1050
     1051
     1052
     1053
     1054
     1055
     1056
     1057
     1058
     1059
     1060
     1061
     1062
     1063
     1064
     1065
     1066
     1067
     1068
     1069
     1070
     1071
     1072
     1073
     1074
     1075
     1076
     1077
     1078
     1079
     1080
     1081
     1082
     1083
     1084
     1085
     1086
     1087
     1088
     1089
     1090
     1091
     1092
     1093
     1094
     1095
     1096
     1097
     1098
     1099
     1100
     1101
     1102
     1103
     1104
     1105
     1106
     1107
     1108
     1109
     1110
     1111
     1112
     1113
     1114
     1115
     1116
     1117
     1118
     1119
     1120
     1121
     1122
     1123
     1124
     1125
     1126
     1127
     1128
     1129
     1130
     1131
     1132
     1133
     1134
     1135
     1136
     1137
     1138
     1139
     1140
     1141
     1142
     1143
     1144
     1145
     1146
     1147
     1148
     1149
     1150
     1151
     1152
     1153
     1154
     1155
     1156
     1157
     1158
     1159
     1160
     1161
     1162
     1163
     1164
     1165
     1166
     1167
     1168
     1169
     1170
     1171
     1172
     1173
     1174
     1175
     1176
     1177
     1178
     1179
     1180
     1181
     1182
     1183
     1184
     1185
     1186
     1187
     1188
     1189
     1190
     1191
     1192
     1193
     1194
     1195
     1196
     1197
     1198
     1199
     1200
     1201
     1202
     1203
     1204
     1205
     1206
     1207
     1208
     1209
     1210
     1211
     1212
     1213
     1214
     1215
     1216
     1217
     1218
     1219
     1220
     1221
     1222
     1223
     1224
     1225
     1226
     1227
     1228
     1229
     1230
     1231
     1232
     1233
     1234
     1235
     1236
     1237
     1238
     1239
     1240
     1241
     1242
     1243
     1244
     1245
     1246
     1247
     1248
     1249
     1250
     1251
     1252
     1253
     1254
     1255
     1256
     1257
     1258
     1259
     1260
     1261
     1262
     1263
     1264
     1265
     1266
     1267
     1268
     1269
     1270
     1271
     1272
     1273
     1274
     1275
     1276
     1277
     1278
     1279
     1280
     1281
     1282
     1283
     1284
     1285
     1286
     1287
     1288
     1289
     1290
     1291
     1292
     1293
     1294
     1295
     1296
     1297
     1298
     1299
     1300
     1301
     1302
     1303
     1304
     1305
     1306
     1307
     1308
     1309
     1310
     1311
     1312
     1313
     1314
     1315
     1316
     1317
     1318
     1319
     1320
     1321
     1322
     1323
     1324
     1325
     1326
     1327
     1328
     1329
     1330
     1331
     1332
     1333
     1334
     1335
     1336
     1337
     1338
     1339
     1340
     1341
     1342
     1343
     1344
     1345
     1346
     1347
     1348
     1349
     1350
     1351
     1352
     1353
     1354
     1355
     1356
     1357
     1358
     1359
     1360
     1361
     1362
     1363
     1364
     1365
     1366
     1367
     1368
     1369
     1370
     1371
     1372
     1373
     1374
     1375
     1376
     1377
     1378
     1379
     1380
     1381
     1382
     1383
     1384
     1385
     1386
     1387
     1388
     1389
     1390
     1391
     1392
     1393
     1394
     1395
     1396
     1397
     1398
     1399
     1400
     1401
     1402
     1403
     1404
     1405
     1406
     1407
     1408
     1409
     1410
     1411
     1412
     1413
     1414
     1415
     1416
     1417
     1418
     1419
     1420
     1421
     1422
     1423
     1424
     1425
     1426
     1427
     1428
     1429
     1430
     1431
     1432
     1433
     1434
     1435
     1436
     1437
     1438
     1439
     1440
     1441
     1442
     1443
     1444
     1445
     1446
     1447
     1448
     1449
     1450
     1451
     1452
     1453
     1454
     1455
     1456
     1457
     1458
     1459
     1460
     1461
     1462
     1463
     1464
     1465
     1466
     1467
     1468
     1469
     1470
     1471
     1472
     1473
     1474
     1475
     1476
     1477
     1478
     1479
     1480
     1481
     1482
     1483
     1484
     1485
     1486
     1487
     1488
     1489
     1490
     1491
     1492
     1493
     1494
     1495
     1496
     1497
     1498
     1499
     1500
     1501
     1502
     1503
     1504
     1505
     1506
     1507
     1508
     1509
     1510
     1511
     1512
     1513
     1514
     1515
     1516
     1517
     1518
     1519
     1520
     1521
     1522
     1523
     1524
     1525
     1526
     1527
     1528
     1529
     1530
     1531
     1532
     1533
     1534
     1535
     1536
     1537
     1538
     1539
     1540
     1541
     1542
     1543
     1544
     1545
     1546
     1547
     1548
     1549
     1550
     1551
     1552
     1553
     1554
     1555
     1556
     1557
     1558
     1559
     1560
     1561
     1562
     1563
     1564
     1565
     1566
     1567
     1568
     1569
     1570
     1571
     1572
     1573
     1574
     1575
     1576
     1577
     1578
     1579
     1580
     1581
     1582
     1583
     1584
     1585
     1586
     1587
     1588
     1589
     1590
     1591
     1592
     1593
     1594
     1595
     1596
     1597
     1598
     1599
     1599
     1600
     1601
     1602
     1603
     1604
     1605
     1606
     1607
     1608
     1609
     1610
     1611
     1612
     1613
     1614
     1615
     1616
     1617
     1618
     1619
     1619
     1620
     1621
     1622
     1623
     1624
     1625
     1626
     1627
     1628
     1629
     1629
     1630
     1631
     1632
     1633
     1634
     1635
     1636
     1637
     1638
     1639
     1639
     1640
     1641
     1642
     1643
     1644
     1645
     1646
     1647
     1648
     1649
     1649
     1650
     1651
     1652
     1653
     1654
     1655
     1656
     1657
     1658
     1659
     1659
     1660
     1661
     1662
     1663
     1664
     1665
     1666
     1667
     1668
     1669
     1669
     1670
     1671
     1672
     1673
     1674
     1675
     1676
     1677
     1678
     1679
     1679
     1680
     1681
     1682
     1683
     1684
     1685
     1686
     1687
     1688
     1689
     1689
     1690
     1691
     1692
     1693
     1694
     1695
     1696
     1697
     1698
     1699
     1699
     1700
     1701
     1702
     1703
     1704
     1705
     1706
     1707
     1708
     1709
     1709
     1710
     1711
     1712
     1713
     1714
     1715
     1716
     1717
     1718
     1719
     1719
     1720
     1721
     1722
     1723
     1724
     1725
     1726
     1727
     1728
     1729
     1729
     1730
     1731
     1732
     1733
     1734
     1735
     1736
     1737
     1738
     1739
     1739
     1740
     1741
     1742
     1743

```

```

fig.update_layout(
    title="Hourly/Weekly Crime Trends",
)
return fig

```

This function gets a bit complicated, but it follows the same general structure as our previous ones. To set up the framing of the subplots, we use the `make_subplots()` function which takes in the number of rows of the plot (just 1) as well as the number of columns (7, one for each day of the week).

We then manipulate and pass in the data required to create each subplot as a `go.Scatter()` object. After this, all `go.Scatter()` objects are appended to our main plot using the `append_trace()` function.

Go ahead and add all of this to `app.py`, and run it to see the final version of our app!

1.10 Conclusions (5 mts)

In this case, we built an interactive dashboard which allows business users to gain insight into crime trends by crime type and by district across time. These insights can be used to mitigate crime in high-incident areas by facilitating increased preparedness by the police department.

Some interesting findings include:

1. Most larceny crimes are committed between 4 - 7 PM on weekdays. There is a sharp decline after 1 - 2 AM on most nights. This pattern is mostly consistent across crime types
2. A few exceptions to this include auto theft, which generally occurs after 8PM, and disorderly conduct, which happens mostly around midnight.

1.11 Takeaways (5 mts)

In this case, you continued leveraging the skills you picked up in the previous case about setting up Dash components and callback functions to dynamically populate them. Additionally, you learned about a few new things in Dash:

1. Using a SQL database as a data source
2. Creating a hierarchical div layout with multiple nested divs
3. Creating subplots within another plot

In future cases, you'll connect to Amazon Web Services (AWS) and link to an in-cloud database as a data source.