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  CAZENOVE 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.cs

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': lab
                                            ),
                                            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.

```
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': labe
                                            ),
                                            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={'textAl
                            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
                                        )
                                }
                        )
                    ])
                ]
            )
        ]
    )
])
```

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]:
```python
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]:
```python
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.

```
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': lab
                                            ),
                                        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={'textAl
                        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=[]
                ),
                html.Div(
                    className='six columns card 2',
                    children=[]
                )
            ]
        )
    ]
  )
])
```

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': labe
                                            ),
                                            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={'textAl
                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 Occurence 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:␣
 ↪"{}-{}".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 numer 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, orientati
        )
    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 Occurence 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': labe
                                    ),
                                    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={'textAl
                    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␣
↪with id date-picker-range and the start_date parameter
          5         dash.dependencies.Input('date-range', 'end_date'),


       NameError: name 'app' is not defined
```

Again, we've outsourced the heavy lifting of this function to a different function `crimes_week()`:

```
[6]: from plotly.subplots import make_subplots
     from pandas.api.types import CategoricalDtype

     def crimes_week(crime_type, start_date, end_date):
         dff = get_filtered_rows(crime_type, start_date, end_date)
         days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
      ↪'Saturday']
         fig = make_subplots(rows=1, cols=7, subplot_titles=days)


         cat_type = CategoricalDtype(categories=days, ordered=True)
         dff['DAY_OF_WEEK'] = dff['DAY_OF_WEEK'].astype(cat_type)

         for crime_name, group in dff.groupby('OFFENSE_CODE_GROUP'):
             i=1
             for day_of_week, week_group in group.groupby('DAY_OF_WEEK'):
                 hour_group = week_group.groupby('HOUR', as_index=False).count()
                 chart = go.Scatter(x=hour_group['HOUR'], y=hour_group['Lat'],␣
      ↪name=crime_name)
                 fig.append_trace(chart, row=1, col=i)
                 i += 1
```

```
    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.