PROJECT GUIDELINES
Dash Uber Rides

Introduction:
Monitoring and reviewing self-driving car trips will become an important task in order to ensure the reliability of the AV system in various conditions and locations. This app lets you visualize Uber trips across New York City and offer various controls for quickly narrowing down on a specific subset of the trips.
Demo: https://dash.gallery/dash-uber-rides-demo/

Step 1: Import the necessary libraries
We will need the given libraries to work with this app

```
import dash
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
import pandas as pd
import numpy as np

import plotly.express as px
from plotly import graph_objs as go
from plotly.graph_objs import *
from datetime import datetime as dt
from datetime import timedelta
from time import strptime
```

Step 2: Initialize the app

```
app = dash.Dash(__name__)

app.title = "New York Uber Rides"

server = app.server
```

Step 3: Fetch the plotly mapbox token

mapbox_access_token =
"pk.eyJ1IjoicGxvdGx5bWFwYm94IiwiYSI6ImNrOWJqb2F4djBnMjEzbG50amg0dnJieG4ifQ.Zme1-Uzoi75IaFbieBDl3A"

Step 4: Read the Uber rides data from the following locations and merge the dataset in one dataframe

https://raw.githubusercontent.com/plotly/datasets/master/uber-rides-data1.csv
https://raw.githubusercontent.com/plotly/datasets/master/uber-rides-data2.csv
https://raw.githubusercontent.com/plotly/datasets/master/uber-rides-data3.csv

Your dataset should look like this after you have finished merging. Then create a list of dataframes split by dates. `dfs = dict(tuple(df.groupby(df['Date/Time'].dt.date)))`

	Date/Time	Lat	Lon
0	2014-04-01 00:11:00	40.769	-73.9549
1	2014-04-01 00:17:00	40.7267	-74.0345
2	2014-04-01 00:21:00	40.7316	-73.9873
3	2014-04-01 00:28:00	40.7588	-73.9776
4	2014-04-01 00:33:00	40.7594	-73.9722
...	...	...	...
1511437	2014-09-30 22:57:00	40.7668	-73.9845
1511438	2014-09-30 22:57:00	40.6911	-74.1773
1511439	2014-09-30 22:58:00	40.8519	-73.9319
1511440	2014-09-30 22:58:00	40.7081	-74.0066
1511441	2014-09-30 22:58:00	40.714	-73.9496

4534327 rows × 3 columns

Step 5: Create a dictionary for all the locations that we want to highlight in our scatter map as follows:

```
# Dictionary of important locations in New York

list_of_locations = {

   "Madison Square Garden": {"lat": 40.7505, "lon": -73.9934},

   "Yankee Stadium": {"lat": 40.8296, "lon": -73.9262},

   "Empire State Building": {"lat": 40.7484, "lon": -73.9857},

   "New York Stock Exchange": {"lat": 40.7069, "lon": -74.0113},

   "JFK Airport": {"lat": 40.644987, "lon": -73.785607},

   "Grand Central Station": {"lat": 40.7527, "lon": -73.9772},

   "Times Square": {"lat": 40.7589, "lon": -73.9851},

   "Columbia University": {"lat": 40.8075, "lon": -73.9626},

   "United Nations HQ": {"lat": 40.7489, "lon": -73.9680},}
```

Step 6: Create a layout sketch for the menu selectors and the dropdown


```
app.layout = html.Div(
   children=[
         # Column for menu selectors
         html.Div(
             className="row",
             children=[

                 html.H2(),
                 html.P(

                     ),
                 html.Div(
                         className="div-for-dropdown",
                         children=[
                             dcc.DatePickerSingle(
                                 id="date-picker",

                             )
                         ],
                     ),
                     # Change to side-by-side for mobile layout
                     html.Div(
                         className="row",
                         children=[
                             html.Div(
                                 className="div-for-dropdown",
                                 children=[
                                     # Dropdown for locations on map
                                     dcc.Dropdown(
                                         id="location-dropdown",

                                     )
                                 ],
                             ),
                             html.Div(
                                 className="div-for-dropdown",
                                 children=[
                                     # Dropdown to select times
                                     dcc.Dropdown(
                                         id="bar-selector",

                                     )
                                 ],
                             ),
                         ],
                     ),
                     html.P(id="total-rides"),
                     html.P(id="total-rides-selection"),
                     html.P(id="date-value"),
             ],
             style={}
         ),
         ]
      )
```

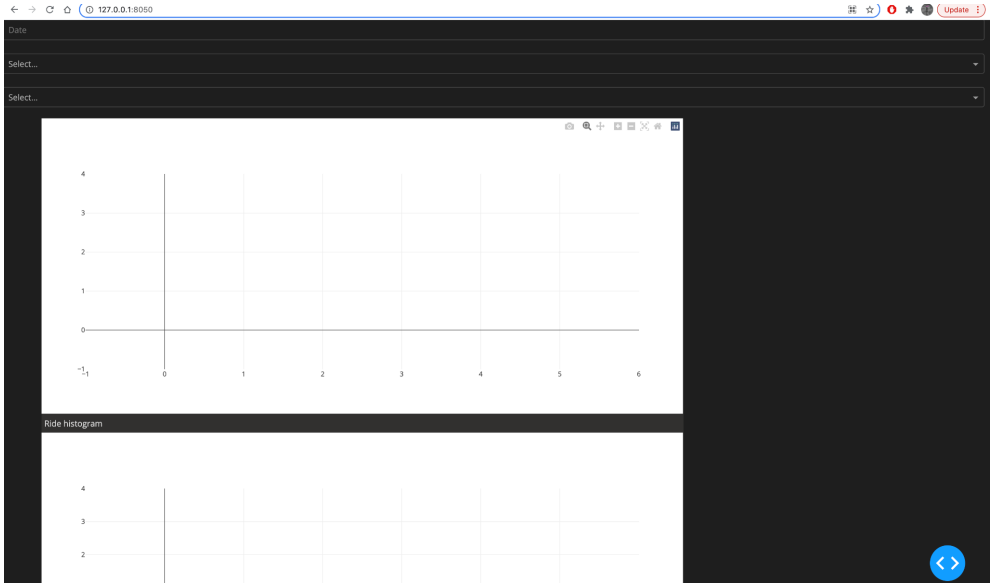
Step 7: Update the layout sketch to add another column for displaying graphs

```
            style={}
        ),

        # Column for app graphs and plots
            html.Div(
                className="eight columns div-for-charts bg-grey",
                children=[
                    dcc.Graph(id="map-graph"),
                    html.Div(
                        className="text-padding",
                        children=[
                            "Ride histogram"
                        ],
                    ),
                    dcc.Graph(id="histogram"),
                ],
                style={

                }
            ),
```

Layout so far



Step 8. Update the core components as per these specifications:

1. **Component: html.H2**
   ```
   Value = "DASH - UBER DATA APP"
   ```

2. **Component: html.P**
   ```
   Value = """Select different days using the date picker or by
   selecting different time frames on the histogram."""
   ```

3. **Component: DatePickerSingle**
   ```
   id="date-picker"
   min_date_allowed=dt(2014, 4, 1)
   max_date_allowed=dt(2014, 9, 30)
   initial_visible_month=dt(2014, 4, 1)
   date=dt(2014, 4, 1).date()
   display_format="MMMM D, YYYY"
   ```

4. **Component: Location Dropdown**
   ```
   id="location-dropdown"
   options=[{"label": i, "value": i} for i in list_of_locations]
   placeholder="Select a location"
   ```

5. **Component: Multi-valued dropdown for hour selector**
   ```
   id="bar-selector"
   options=[{"label": str(n) + ":00", "value": str(n),}  for n in
   range(24) ]
   multi=True
   placeholder="Select certain hours"
   ```

6. **Style for Menu Selector Div Column**
   ```
   "border": "0px solid black",
   "padding-top": "12px",
   "padding-bottom": "12px",
   "float": "left",
   "width": "30%"
   ```

7. **Style for Graph Div Column**
   ```
   "display": "flex",
   "flex-direction": "column",
   "height": "80vh",
   "width": "60%"
   ```

Step 9. Create a callback function to update scatter map box

1. **Function name:** update_scattermap

2. **Input to callback function:**
- Input("date-picker", "date") ->> datePicked
- Input("bar-selector", "value") ->> selectedData
- Input("location-dropdown", "value") ->> selectedLocation

3. **Output of callback function:**
- Output("map-graph", "figure")

4. **Initial Values:**
- zoom = 12.0
- latInitial = 40.7272
- lonInitial = -73.991251
- bearing = 0

5. **If a location is selected, update the initial values as follows:**
- zoom = 15.0
- latInitial = list_of_locations[selectedLocation]["lat"]
- lonInitial = list_of_locations[selectedLocation]["lon"]

6. **Format the value of 'datePicked':** date_picked = dt.strptime(datePicked, "%Y-%m-%d")

7. **Create a new dataframe df_day given the input variable 'datePicked':**
   df_day = dfs[date_picked.date()]

8. **If the filtered dataframe is not empty, add another column 'hour', derived from 'Date/Time' column**

```
if df_day.shape[0] > 0:
        df_day['hour'] = df_day["Date/Time"].dt.hour
```

9. **If user input for hour is present, filter the df_day dataframe for the selected hours in a new dataframe 'df_hour'**

```
if df_day.shape[0] > 0:
        df_day['hour'] = df_day["Date/Time"].dt.hour
         if selectedData != None:
                df_hour = df_day[df_day['hour'].isin(selectedData)]
        else:
                df_hour = df_day
         df_hour.index = df_hour["Date/Time"]
```

10. Create a graph object using the following information

Data: List of 2 scattermapbox:
https://plotly.github.io/plotly.py-docs/generated/plotly.express.scatter_mapbox.html,
https://plotly.com/python-api-reference/generated/plotly.graph_objects.Scattermapbox.html

	Scattermapbox - 1	**Scattermapbox - 2**
lat	df_hour["Lat"]	[list_of_locations[i]["lat"] for i in list_of_locations]
lon	df_hour["Lon"]	[list_of_locations[i]["lon"] for i in list_of_locations]
mode	"markers"	"markers"
hoverinfo	"lat+lon+text"	"text"
text	df_hour.index.hour	[i for i in list_of_locations]
marker	dict(showscale=True, color=np.append(np.insert(df_hour.index.hour, 0, 0), 23), opacity=0.5, size=5, colorscale=[[0, "#F4EC15"], [0.04167, "#DAF017"], [0.0833, "#BBEC19"], [0.125, "#9DE81B"], [0.1667, "#80E41D"], [0.2083, "#66E01F"], [0.25, "#4CDC20"], [0.292, "#34D822"], [0.333, "#24D249"], [0.375, "#25D042"], [0.4167, "#26CC58"], [0.4583, "#28C86D"], [0.50, "#29C481"], [0.54167, "#2AC093"], [0.5833, "#2BBCA4"], [1.0, "#613099"],], colorbar=dict(title="Time of Day", x=0.93, xpad=0, nticks=24, tickfont=dict(color="#d8d8d8"), titlefont=dict(color="#d8d8d8"), thicknessmode="pixels",),)	dict(size=8, color="#ffa0a0")

Layout:

- **autosize**=True,
- **margin**=go.layout.Margin(l=0, r=35, t=0, b=0),
- **showlegend**=False,
- **mapbox**=dict(
    accesstoken=mapbox_access_token,
    center=dict(lat=latInitial, lon=lonInitial), # 40.7272 # -73.991251
    style="dark",
    bearing=bearing,
    zoom=zoom,
  ),
- **updatemenus**=[
    dict(
      buttons=(
        [
          dict(
            args=[
              {
                "mapbox.zoom": 12,
                "mapbox.center.lon": "-73.991251",
                "mapbox.center.lat": "40.7272",
                "mapbox.bearing": 0,
                "mapbox.style": "dark",
              }
            ],
            label="Reset Zoom",
            method="relayout",
)
        ]
      ),
      direction="left",
      pad={"r": 0, "t": 0, "b": 0, "l": 0},
      showactive=False,
      type="buttons",
      x=0.45,
      y=0.02,
      xanchor="left",
      yanchor="bottom",
      bgcolor="#323130",
      borderwidth=1,
      bordercolor="#6d6d6d",
      font=dict(color="#FFFFFF"),
)
  ],
),
)

Step 10. Create a callback function to update the total number of rides tag

1. **Function name: update_total_rides**
2. **Input to callback function:**

- Input("date-picker", "date") ->> datePicked
- Input("location-dropdown", "value") ->> selectedLocation

3. **Output of callback function:**

- Output("total-rides", "children")

4. **Select and return the dataframe from the dictionary(created in Step 4) by date**

```
return "Total Number of rides: {:,d}".format(
        (dfs[date_picked.date()]).shape[0]
```

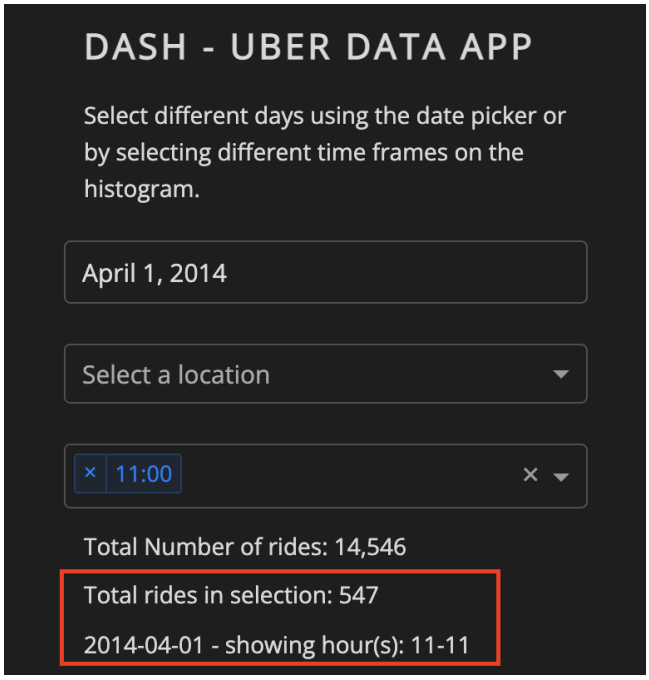
Step 11. Create a callback function to update the total number of rides by selected hours (inclusive of all locations)

1. **Function name: update_total_rides**
2. **Input to callback function:**

- Input("date-picker", "date") ->> datePicked
- Input("bar-selector", "value") ->> selection

3. **Output of callback function:**

- Output("total-rides-selection", "children")
- Output("date-value", "children")



Step 12. Create a callback function to update the histogram plot. This function should also update the multi-valued dropdown for hours when a bar on the histogram is clicked. (Graph Interactivity and Cross Filtering)

1. **Function name: update_histogram**
2. **Input to callback function:**

- Input("date-picker", "date")
- Input("bar-selector", "value")
- Input('histogram', 'clickData')

3. **Output to callback function:**

- Output("histogram", "figure"),
- Output("bar-selector", "value")

4. **Format the input date picker to match datetime format:**
   date_picked = dt.strptime(datePicked, "%Y-%m-%d")
5. **Create a dataframe 'df_day' by selecting the dataframe from the dictionary created in Step 4 by date:** df_day = dfs[date_picked.date()]
6. **Rollup the 'Date/Time' column to represent data only by date and hours. The minutes and seconds should be 00:00:**
   df_day['Date/Time'] = df_day['Date/Time'].astype('datetime64[h]')
7. **Create a histogram with plotly express histogram. Conditions for histogram:**
   - The bars on the histogram should be colored by the hours in the 'Date/Time' column. Try various properties of plotly express to update the histogram bars color_discrete_sequence= px.colors.sequential.Viridis_r color_discrete_map=colorVal where
     - colorVal = {0:"#F4EC15",1:"#DAF017",2:"#BBEC19",3:"#9DE81B",4:"#80E41D",5:"#66E01F",6:"#4CDC20",7:"#34D822",8:"#24D249",9:"#25D042",10:"#26CC58",11:"#28C86D",12:"#29C481",13:"#2AC093",14:"#2BBCA4",15:"#2BB5B8",16:"#2C99B4",17:"#2D7EB0",18:"#2D65AC",19:"#2E4EA4",20:"#2E38A4",21:"#3B2FA0",22:"#4E2F9C",23:"#603099"}
   - Any hour selected by user in the dropdown should be highlighted as white bar in the histogram
   - Any bar selected on the histogram should also be highlighted as white when clicked by the user. The hours for the selected bar should update the multi-valued dropdown for hour selection