Dash(board) of Norwegian Flight Data

Ashish Kumar, Irbaz Bin Riaz, Ramakrishna Devarakonda, Craig Fulgency, Justin Greenberg

Idea and Chosen Frameworks

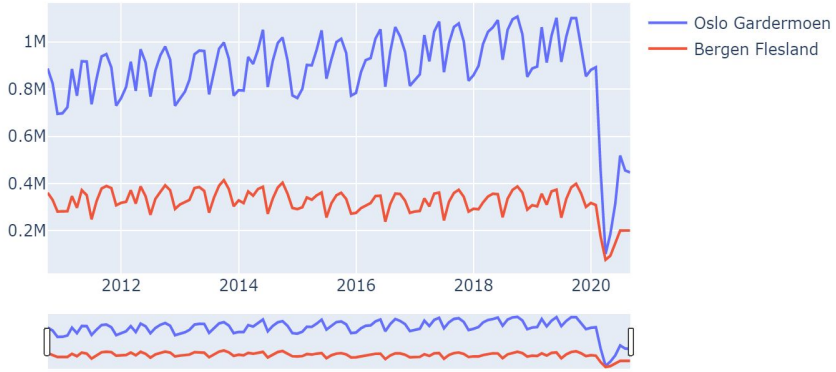
- We wanted to pick geo mapping as our prime visualization and that drew us towards airports
  - Norway's airports transposed over the country's geography would look nice, dataset would have 52 airports, country with an extensive network of domestic travel.
- We found the raw data from the Norway Statistics website : https://www.ssb.no/en/statbank/table/08507/
- The frameworks used to depict the data : **Dash, Plotly, Mapbox studio, Pandas, NumPy, Bootstrap,HTML, Jupyter Notebook** and **PostgreSQL**
  - Why Dash?

Dash Open Source

Plotly stewards Python's leading data viz and UI libraries.

Data Munging and Loading

- Read in downloaded csv, df.sort_values(by=”airport”)
- Renamed columns and identified missing values
- Added geo data for each airport by merging another df
  - Manually added some missing data
- Created df_melt group by passengers
- Convert data format YYYY/MM/DD



- Testing on data, plotted scatter graph etc. within .ipynb
- Loaded final DF into POSTGRESQL DB using psycopg2 python driver
- Once the data load is done we pulled it back to data frame.
  - Assure interaction between front and back end

SELECT * FROM passenger_data;

Data Output | Explain | Messages | Notifications

	id [PK] integer	airport character varying (40)	type_of_traffic character varying (40)	location character varying (470)	latitude double precision	longitude double precision	date date	passengers integer
1	0	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-04...	4221
2	1	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-05...	6229
3	2	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-06...	13248
4	3	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-03...	16487
5	4	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-09...	19571
6	5	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2020-08...	20621
7	6	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2012-12...	22131
8	7	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2010-12...	22914
9	8	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2013-12...	22941
10	9	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2015-01...	23040
11	10	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2014-01...	23105
12	11	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2011-01...	23369
13	12	Alta	All commercial flights	Alta, Troms og Finnmark, Norge	70.04962755	23.08254009804839	2011-02...	23484

Application Prep

```python
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from itertools import cycle
from dash.dependencies import Input,Output
from dash_bootstrap_components._components.Navbar import Navbar
import dash
import dash_table
import dash_bootstrap_components as dbc
import dash_core_components as dcc
import dash_html_components as html
from DB.connect import connect
from DB.postgresql import postgresql_to_dataframe
### import prepared data
```

Required packages for DASH app

```python
app = dash.Dash(__name__,external_stylesheets=[dbc.themes.BOOTSTRAP])
```

Bootstrap stylesheet initialization

```python
# Connect to the database
conn = connect()
column_names = ["id", "airport", "type of traffic", "location","latitude","longitude",'date','passengers']
# Execute the "SELECT *" query
df_melt = postgresql_to_dataframe(conn, "select * from passenger_data", column_names)
conn.close()
print('connection closed')
```

Read backend PostgreSQL into df

```python
#---------------------------------------------------------------
# Define dropdowns
#---------------------------------------------------------------
traffic_dropdown = dcc.Dropdown(
    id="traffic-dropdown",
    options=[
        {"label": t, "value": t} for t in type_of_traffic
    ],
    value="All commercial flights",
    clearable=False,
    multi=False,
    style={"width": "100%"},
)
```

```python
# ---------------------------------------------------------------
# Define buttons
# ---------------------------------------------------------------
year_set_btn = dbc.ButtonGroup(
    [
        dbc.Button(
            "Select all",
            id="year-btn-all",
            # outline=True,
            color="primary",
            className="mr-1",
        ),
        dbc.Button(
            "Deselect",
            id="year-btn-none",
            # outline=True,
            color="primary",
            className="mr-1",
        ),
    ],
    id="year-set-btn",
    size="md",
)
```

Define search bar, navbar, buttons, dropdowns using DBC + defined graphs (line, bar)

Application Prep

Created content container using dash Bootstrap card component to hold multiple dropdowns and buttons

```
# ------------------------------------------------------------------------------
# Define overview options card
# ------------------------------------------------------------------------------
overview_options_card = dbc.Card(
    [
        dbc.Row(
            [
                dbc.Col(
                    [
                        dbc.Row([dbc.Label("Select scale")]),
                        dbc.Row([scale_dropdown]),
                        html.Br(),
                        dbc.Row([dbc.Label("Select type of traffic")]),
                        dbc.Row([traffic_dropdown]),
                        html.Br(),
                        dbc.Row([dbc.Label("Search and select years")]),
                        dbc.Row([year_set_btn]),
                        dbc.Row([year_dropdown]),
                        html.Br(),
                        dbc.Row([dbc.Label("Search and select months")]),
                        dbc.Row([month_set_btn]),
                        dbc.Row([month_dropdown]),
                        html.Br(),
                        dbc.Row([dbc.Label("Search and select airports")]),
                        dbc.Row([airport_set_btn]),
                        dbc.Row([airport_dropdown]),
                    ], style={"width": "100%"},
                )
            ], style={"width": "100%"},
        ),
    ],
    body=True,
    style={"width": "100%"},
)
```

Application Prep

```
# ------------------------------------------------------------------------------
# Define tabs
# ------------------------------------------------------------------------------
tab1_content = dbc.Row(
    [
        html.Div([
            html.Br(),
            html.Span('Airport Passenger Amount by Location', style={
                    "font-size": 22, "color": color_2, 'font-weight': 'bold'}),
            html.Br(),
            html.Span('Graphical representation of the amount of passenger in Nc
                    "font-size": 14, "color": color_2}),
        ]
        ),
        graph_map,
    ],
    no_gutters=True,
)
```

```
tabs = dbc.Tabs(
    [
        dbc.Tab(tab1_content, tab_id="tab_map", label="Geographical"),  # style={"width": "100%"}),
        dbc.Tab(tab2_content, tab_id="tab_total", label="Categorical"),  # style={"width": "100%"})
        dbc.Tab(tab3_content, tab_id="tab_time", label="Time"),  # style={"width": "100%"}),
        dbc.Tab(tab4_content, tab_id="tab_table", label="Table"),  # style={"width": "100%"}),
    ],
    id="tabs",
    active_tab="tab_map",
    style={"width": "100%"}
    # style={"height": "auto", "width": "auto"},
)
```

- Our application requires 4 tabs to hold 4 different visualizations
  - created tab content first and then assigned to tab functions

Application Prep

Defined the default UI look for each tab using callback function

```
# ------------------------------------------------------------
# Define callback to toggle tabs
# ------------------------------------------------------------
@app.callback(
    [
        Output("scale-dropdown",'disabled'),
        Output("menu_1", "is_open"),
        Output("menu_col_1", "width"),
        Output("menu_col_1", "xs"),
        Output("menu_col_1", "sm"),
        Output("menu_col_1", "md"),
        Output("menu_col_1", "lg"),
        Output("menu_col_1", "xl")
    ],
        Input("tabs", "active_tab"),
)
def toggle_tabs(tab_id):
    if(tab_id == 'tab_time' or tab_id == 'tab_table'):
        return False,False,'0%',0,0,0,0,0
    elif tab_id == 'tab_map':
        return True,True,'0%',6,5,4,3,2
    elif tab_id == 'tab_total':
        return False,True,'0%',6,5,4,3,2
```

Application Prep

- Defined table using dash component dash_table
- Table was designed with sort, custom cell styles and custom data styles
  - Pagination by default

```
# ------------------------------------------------------------------------------
# Define table
# ------------------------------------------------------------------------------
table = dash_table.DataTable(
    id='table',
    columns = [{"name": i, "id": i} for i in df_table.columns],
    data = df_table.to_dict('records'),
    filter_action='native',
    sort_action='native',
    page_size=25,
    style_cell={
        'overflow': 'hidden',
        'textOverflow': 'ellipsis',
        'maxWidth': 0,
        # 'textAlign': 'left',
    },
    style_cell_conditional=[
        {'if': {'column_id': 'airport'},
         'width': '18%', 'textAlign': 'left'},
        {'if': {'column_id': 'type of traffic'},
         'width': '17%', 'textAlign': 'left'},
        {'if': {'column_id': 'location'},
         'width': '32%', 'textAlign': 'left'},
        {'if': {'column_id': 'latitude'},
         'width': '5%'},
        {'if': {'column_id': 'longitude'},
         'width': '5%'},
        {'if': {'column_id': 'date'},
         'width': '7%'},
        {'if': {'column_id': 'passengers'},
         'width': '6%'},
    ],
    style_data_conditional=[
        {
            'if': {'row_index': 'odd'},
            'backgroundColor': 'rgb(248, 248, 248)'
        }
    ],
    style_header={
        'backgroundColor': 'rgb(230, 230, 230)',
        'fontWeight': 'bold'
    },
```

Application Layout

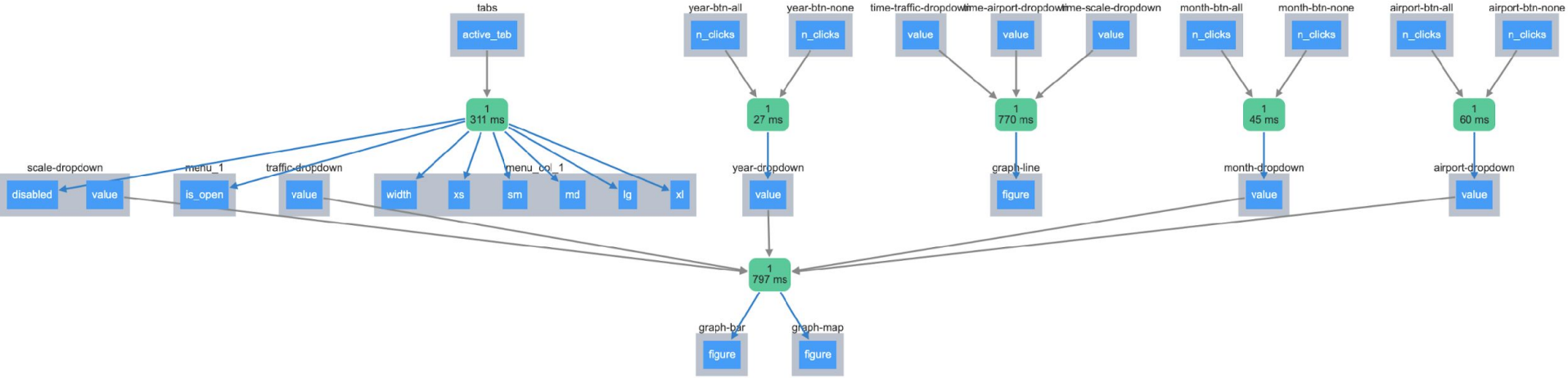
Using dbc to define layout based on which tab is clicked

```
# ------------------------------------------------------------------------------
# Define layout
# ------------------------------------------------------------------------------
app.layout = html.Div(
    [
        navbar,
        dbc.Row(
            [
                dbc.Col(
                    [
                        dbc.Collapse(
                            overview_options_card,
                            id="menu_1",
                        )
                    ], id="menu_col_1", width=6, xs=6, sm=5, md=4, lg=3, xl=2
                ),
                dbc.Col([tabs]),
            ], style={"height": "auto", "width": "99%"},
        )
    ],
    # style={"height": "auto", "width": "auto"},
)
```

Data Visualization

- Hinges on callback functionality in Dash & Plotly
- Map (**Mapbox Studio**)
- Bar Graph
- Time Chart
- Table made with **dash_table**
  - Pagination
  - Striped cell style

Our callback function flow



Challenges

- Loading
  - Complete dataset was confusing, working with sums
    - Had to carefully select from website which data to download
  - Adding unique CSV delimiters because columns had commas in them
  - Encoding is **ISO-8859-1** because of Norwegian characters
  - Credentials coming from .ini file
    - Systematic connection with .py functions used for PostgreSQL
- Dash
  - Organizing callback functions for when an input is received
  - No index.html or CSS, but bootstrap components are used

```
import csv
import psycopg2
from DB.connect import connect
conn = connect()
cur = conn.cursor()
with open('cool_df.csv', 'r') as f:
    next(f)
    cur.copy_from(f, 'passenger_data', sep='|')
    conn.commit()
    print('table load is done')

conn.close()
print('connection closed')
```