**Dashboard de Medium**

# **Airplane Flight Reports — Python Dashboard with Dash and Plotly**



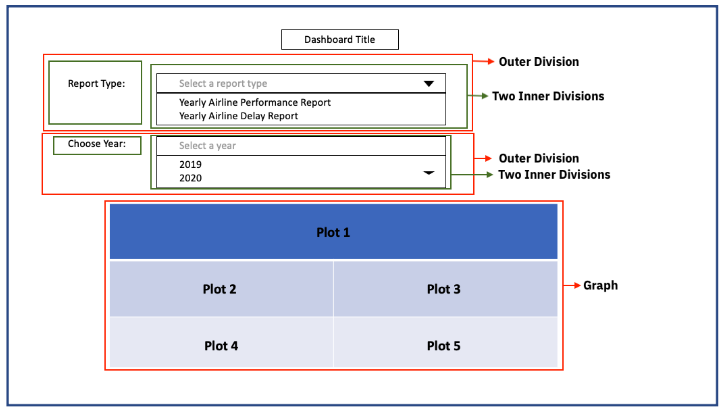
Image by [Lars\_Nissen](https://pixabay.com/users/lars_nissen-2780243/?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=3702676" \t "_blank) from [Pixabay](https://pixabay.com/?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=3702676" \t "_blank)

Dash- as a python framework is used to build interactive web application to enrich experience of data visualization. In this project, I will explain how to create a dash application to present an interactive report. The aim of this project is to create an interactive visualization to monitor and report US domestic airline flights performance. This visualization can be used to support decisions to improve flight the reliability thereby increasing customer reliability.

There are two report keys in this project, range in the span of 15 years from 2005 to 2020. The **Yearly Airline Performance Report** consists of five charts as follows :

* Number of flights under different cancellation categories in bar chart.
* Average flight time by reporting airline in line chart.
* Percentage of diverted airport landings per reporting airline in pie chart.
* Number of flights flying from each state in choropleth map.
* Number of flights flying to each state from each reporting airline in treemap chart.

Meanwhile, **Yearly Average Flight Delays Statistics** will provide five line charts that illustrate the average of delay by reporting airline per year caused by : carrier, weather, national air system (NAS), security and late aircraft. The outline of the visualization will look like this :



Visualization layout

This visualization will contain two dropdown menus for choosing report type and year. Each dropdown will be designed with two inner divisions, one containing information about the dropdown and the dropdown itself. The graphs will be shown below the dropdown and to create them I will use callback function to compute the data, create graph and return the layout.

The steps I followed to finish this project :

**Import libraries and data preparation**

Firstly, I imported the required libraries. I used pandas to manipulate data, for the visualization I used plotly, and dash to create an interactive web based dashboard :

import libraries

After importing all the libraries, I loaded the data and made some adjusments. The dataframe contains 110 columns, and I will only use the relevant columns for the visualization: Month, Year, Flights, CancellationCode, Reporting\_Airline, AirTime, DivAirportLandings, OriginState, DestState, CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, and LateAircraftDelay. Then, I defined the year in year\_list from 2005 to 2021.

Prepare the data for each dropdowns using pandas. Here, I used function compute\_data\_choice\_1 for the performance report, and compute\_data\_choice\_2 for the delay report. The outputs are variables which can be use in the next function after I finished the dashboard skeleton. There are 10 variables in total : bar\_data, line\_data, div\_data, map\_data, and tree\_datafor performance report; then avg\_data, avg\_weather, avg\_NAS, avg\_sec, and avg\_latefor the delay report.

data preparation

**Create dash application and get the lab skeleton**

The dash application skeleton consists of **three main components** : titles for the application and dropdowns; dropdown components; and computer graphs that will be updated after callback. I also need to create dash app and suppress callback until all the processes finished :

create dash app

**Update the layout components**

I set dash application main title using html.H1() in the puter division, and html.H2() for the title in the inner divisions. Then, to create dropdown, I putdcc.Dropdown() for each inner division and named the id as 'input-type' and 'input-year'.

Options in the dropdown was written as dictionary using parameter options and given label and value.placeholder= is used to display the name when dropdown in default position. To add titles and adjust styling use parameter style. The code will look like this :

Next, for the graph section, I used empty division to show graph once the input was given. The graph was divided into three divisions and the plots were defined using id :

updatable graph

**Add the application callback function**

We can display outputs based on inputs and update them on real-time dashboard using callback function. The first step in this process is defining the inputs. This input is required to run the computation I already prepared as compute\_data\_choice variables.

The concept of callback is to use type of report and year as input then update the graphs according to these inputs. To create this callback, firstly I defined the callback decorator.

app callback to call the components

There are three components in this @app.callback : output, input, and state. There will be five plots for each input labeled as component\_idwith ‘plot1’ until ‘plot5’ and the plot will be displayed as component\_propertyin childrenelement of html.Div[]. Then, I defined the input as before- by addinginput-type and input-year using property of value. State is an addition to avoid the input from being fired before all the inputs are finished.

**Update the callback function**

After finished the the callback decorator, next I need to define the functions required to call all the graphs and put its proper place. Using function get\_graph(), I wrote all the codes to define the data (df) by year (input-year) then put them into calculation I have prepared (compute\_data\_choice) then form them into graphs.

The graphs displayed will be referring to value stated in dropdowns, thus I also used conditional in the function. The set of graphs displayed by value ‘OPT1’ (Performance Report) will be different with set of graphs by value ‘OPT2’ (Delay Report). The function will look like this :

Lastly, I used dcc.Graph(figure = graph-name) as a component to display the graphs and put them in the right position of dash app skeleton. I also wrote them in order to ensure the five graphs will be displayed according to the layout design.

**Run the application!**

Finally, I have come to the last part of the dashboard creation : Running the application. Just add this code below and run the application to show the port where we can see the dashboard :

After running and get the port, we can run the dashboard page :

