**Let's start creating dash application**

**Theme**

Analyze flight delays in a dashboard.

**Dashboard Components**

* Monthly average carrier delay by reporting airline for the given year.
* Monthly average weather delay by reporting airline for the given year.
* Monthly average national air system delay by reporting airline for the given year.
* Monthly average security delay by reporting airline for the given year.
* Monthly average late aircraft delay by reporting airline for the given year.

NOTE: Year range should be between 2010 and 2020

**Expected Output**

Below is the expected result from the lab. Our dashboard application consists of three components:

* Title of the application
* Component to enter input year
* 5 Charts conveying the different types of flight delay. Chart section is divided into three segments.
  + Carrier and Weather delay in the first segment
  + National air system and Security delay in the second segment
  + Late aircraft delay in the third segment

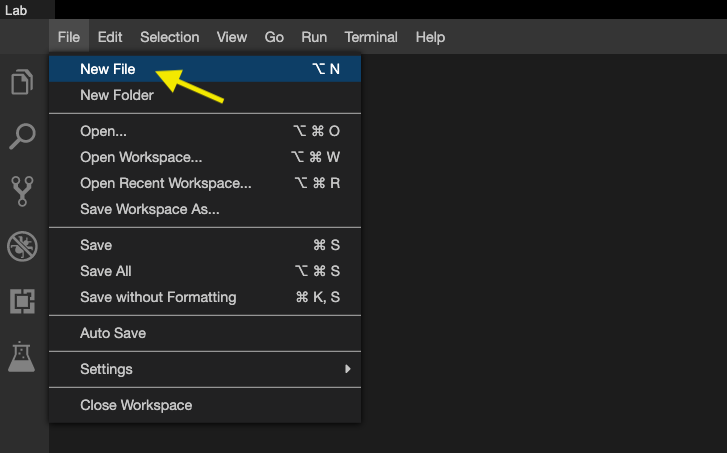


**To do:**

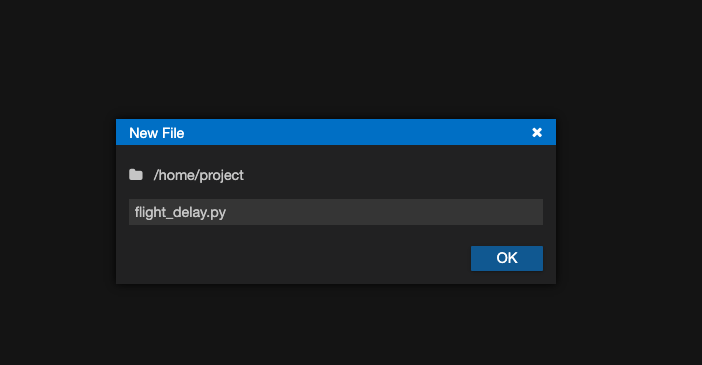
* Design layout for the application.
* Create a callback function. Add callback decorator, define inputs and outputs.
* Review the helper function that performs computation on the provided inputs.
* Create 5 line graphs.
* Run the application.

**Get the tool ready**

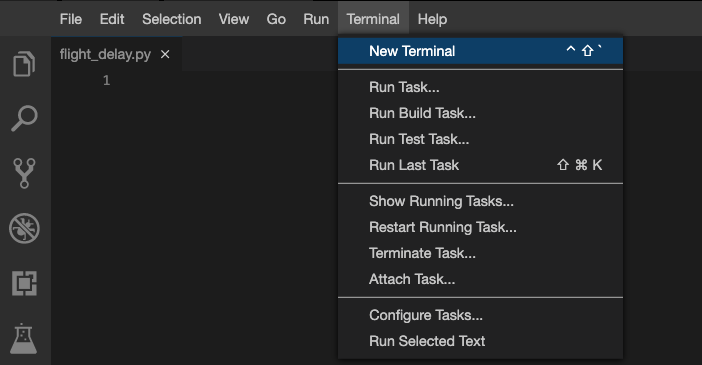
* Create a new python script, by clicking on the menu bar and selecting **File**->**New File**, as in the image below.



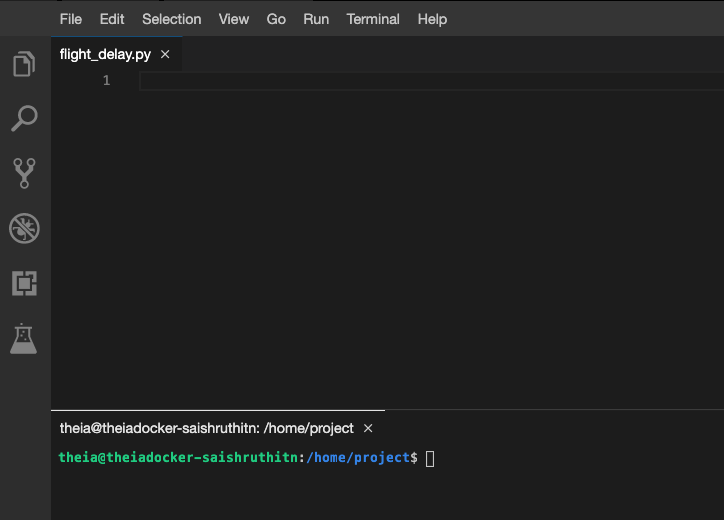
* Provide the file name as flight\_details.py



* Open a new terminal, by clicking on the menu bar and selecting **Terminal**->**New Terminal**, as in the image below.

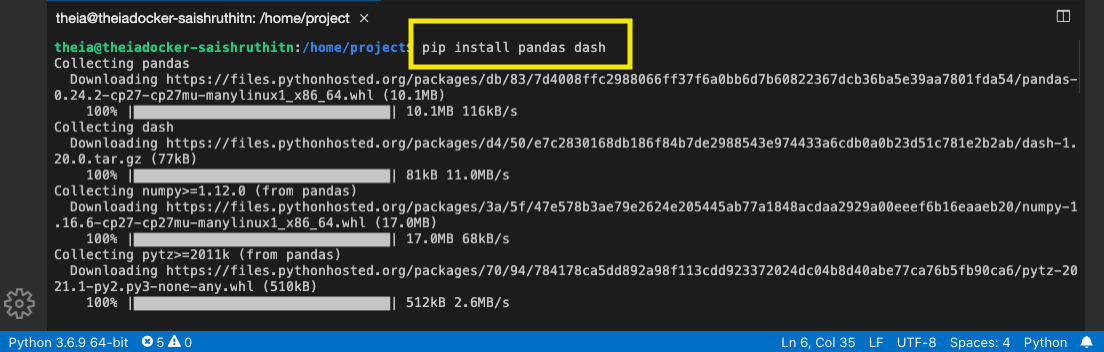


* Now, you have script and terminal ready to start the lab.



* Install python packages required to run the application. Copy and paste the below command to the terminal.

**pip3** install pandas dash



**TASK 1 - Read the data**

Let's start with

* Importing necessary libraries
* Reading the data

Copy the below code to the flight\_delay.py script and review the code.

# Import required libraries

**import** pandas **as** pd

**import** plotly.graph\_objects **as** go

**import** dash

**import** dash\_html\_components **as** html

**import** dash\_core\_components **as** dcc

**from** dash.dependencies **import** Input, Output

# Read the airline data into pandas dataframe

airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',

encoding = "ISO-8859-1",

dtype={'Div1Airport': str, 'Div1TailNum': str,

'Div2Airport': str, 'Div2TailNum': str})

**TASK 2 - Create dash application and get the layout skeleton**

Next, we create a skeleton for our dash application. Our dashboard application layout has three components as seen before:

* Title of the application
* Component to enter input year inside a layout division
* 5 Charts conveying the different types of flight delay

Mapping to the respective Dash HTML tags:

* Title added using html.H1() tag
* Layout division added using html.Div() and input component added using dcc.Input() tag inside the layout division.
* 5 charts split into three segments. Each segment has a layout division added using html.Div() and chart added using dcc.Graph() tag inside the layout division.

Copy the below code to the flight\_delay.py script and review the structure.

*NOTE*: Copy below the current code

# Create a dash application

app = dash.Dash(\_\_name\_\_)

# Build dash app layout

app.layout = html.Div(children=[ html.H1(),

html.Div(["Input Year: ", dcc.Input()],

style={'font-size': 30}),

html.Br(),

html.Br(),

html.Div([

html.Div(),

html.Div()

], style={'display': 'flex'}),

html.Div([

html.Div(),

html.Div()

], style={'display': 'flex'}),

html.Div(, style={'width':'65%'})

])

*NOTE*: We are using display as flex for two outer divisions to get graphs side by side in a row.

# TASK 3 - Update layout components

### Application title

* Title as Flight Delay Time Statistics, align text as center, color as #503D36, and font size as 30.

### Input component

* Update [**dcc.Input**](https://dash.plotly.com/dash-core-components/input?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as input-year, default value as 2010, and type as number. Use style parameter and assign height of the input box to be 35px and font-size to be 30.

### Output component - Segment 1

Segment 1 is the first html.Div(). We have two inner division where first two graphs will be placed.

#### Skeleton

html.Div([

html.Div(),

html.Div()

], style={'display': 'flex'}),

#### First inner division

* Add dcc.Graph() component.
* Update [**dcc.Graph**](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as carrier-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [**dcc.Graph**](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as weather-plot.

### Output component - Segment 2

Segment 2 is the second html.Div(). We have two inner division where the next two graphs will be placed.

#### Skeleon

html.Div([

html.Div(),

html.Div()

], style={'display': 'flex'}),

#### First inner division

* Add dcc.Graph() component.
* Update [**dcc.Graph**](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as nas-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [**dcc.Graph**](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as security-plot.

### Output component - Segment 3

Segment 3 is the last html.Div().

#### Skeleon

html.Div(, style={'width':'65%'})

* Add dcc.Graph() component to the first inner division.
* Update [**dcc.Graph**](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) component id as late-plot.

# TASK 4 - Review and add supporting function

Below is the function that gets input year and data, perform computation for creating charts and plots.

Copy the below code to the flight\_delay.py script and review the structure.

NOTE: Copy below the current code

""" Compute\_info function description

This function takes in airline data and selected year as an input and performs computation for creating charts and plots.

Arguments:

airline\_data: Input airline data.

entered\_year: Input year for which computation needs to be performed.

Returns:

Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.

"""

**def** **compute\_info**(airline\_data, entered\_year):

# Select data

df = airline\_data[airline\_data['Year']==int(entered\_year)]

# Compute delay averages

avg\_car = df.groupby(['Month','Reporting\_Airline'])['CarrierDelay'].mean().reset\_index()

avg\_weather = df.groupby(['Month','Reporting\_Airline'])['WeatherDelay'].mean().reset\_index()

avg\_NAS = df.groupby(['Month','Reporting\_Airline'])['NASDelay'].mean().reset\_index()

avg\_sec = df.groupby(['Month','Reporting\_Airline'])['SecurityDelay'].mean().reset\_index()

avg\_late = df.groupby(['Month','Reporting\_Airline'])['LateAircraftDelay'].mean().reset\_index()

**return** avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late

**TASK 5 - Add the application callback function**

The core idea of this application is to get year as user input and update the dashboard in real-time. We will be using callback function for the same.

Steps:

* Define the callback decorator
* Define the callback function that uses the input provided to perform the computation
* Create graph and return it as an output
* Run the application

Copy the below code to the flight\_delay.py script and review the structure.

*NOTE*: Copy below the current code

# Callback decorator

@app.callback( [

Output(component\_id='carrier-plot', component\_property='figure'),

---

---

---

---

],

Input(....))

# Computation to callback function and return graph

**def** **get\_graph**(entered\_year):

# Compute required information for creating graph from the data

avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_info(airline\_data, entered\_year)

# Line plot for carrier delay

carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrier delay time (minutes) by airline')

# Line plot for weather delay

weather\_fig = ------

# Line plot for nas delay

nas\_fig = ------

# Line plot for security delay

sec\_fig = ------

# Line plot for late aircraft delay

late\_fig = ------

**return**[carrier\_fig, weather\_fig, nas\_fig, sec\_fig, late\_fig]

# Run the app

**if** \_\_name\_\_ == '\_\_main\_\_':

app.run\_server

**TASK 6 - Update the callback function**

**Callback decorator**

* Refer examples provided [**here**](https://dash.plotly.com/basic-callbacks?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01)
* We have 5 output components added in a list. Update output component id parameter with the ids provided in the dcc.Graph() component and set the component property as figure. One sample has been added to the skeleton.
* Update input component id parameter with the id provided in the dcc.Input() component and component property as value.

**Callback function**

Next is to update the get\_graph function. We have already added a function compute\_info that will perform computation on the data using the input.

Mapping the returned value from the function compute\_info to graph:

* avg\_car - input for carrier delay
* avg\_weather - input for weather delay
* avg\_NAS - input for NAS delay
* avg\_sec - input for security delay
* avg\_late - input for late aircraft delay

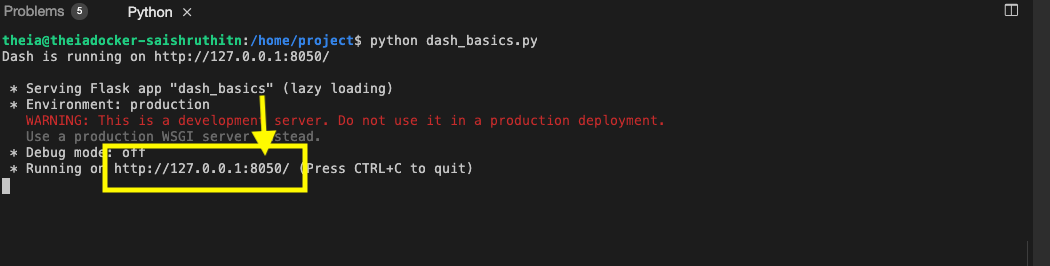
Code has been provided for plotting carrier delay. Follow the same process and use the above mapping to get plots for other 4 delays.

**TASK 6 - Run the application**

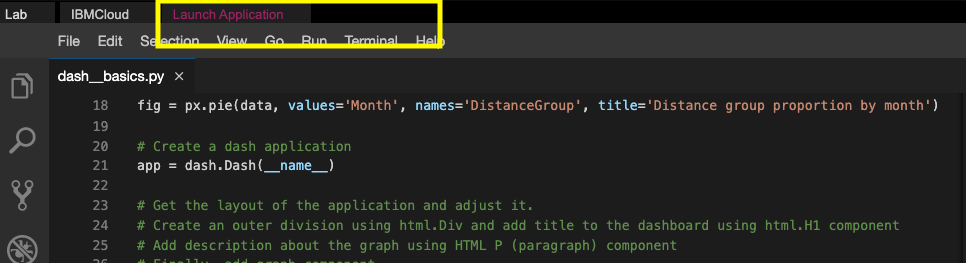
* Copy and paste the below command in the terminal to run the application.

**python3** **flight\_delay**.py

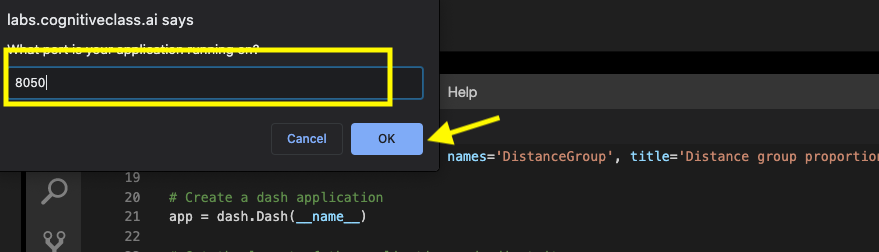
* Observe the port number shown in the terminal.



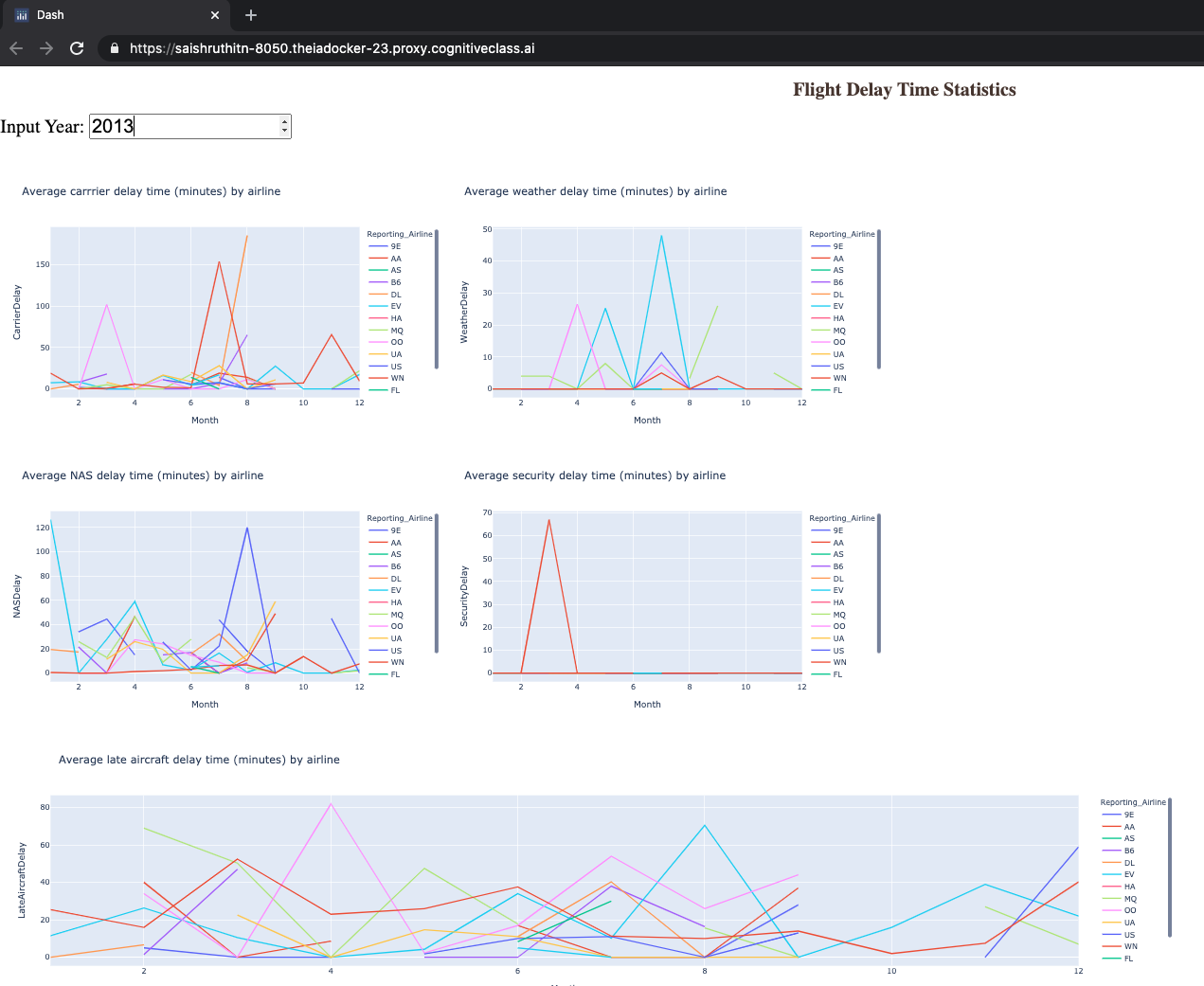
* Click on the Launch Application option from the menu bar.



* Provide the port number and click OK



The app will open in a new browser tab like below:



**Congratulations, you have successfully created your dash application!**

# Import required libraries

import pandas as pd

import dash

import dash\_html\_components as html

import dash\_core\_components as dcc

from dash.dependencies import Input, Output

import plotly.express as px

# Read the airline data into pandas dataframe

airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',

encoding = "ISO-8859-1",

dtype={'Div1Airport': str, 'Div1TailNum': str,

'Div2Airport': str, 'Div2TailNum': str})

# Create a dash application

app = dash.Dash(\_\_name\_\_)

# Build dash app layout

app.layout = html.Div(children=[ html.H1('Flight Delay Time Statistics',

style={'textAlign': 'center', 'color': '#503D36',

'font-size': 30}),

html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',

type='number', style={'height':'35px', 'font-size': 30}),],

style={'font-size': 30}),

html.Br(),

html.Br(),

# Segment 1

html.Div([

html.Div(dcc.Graph(id='carrier-plot')),

html.Div(dcc.Graph(id='weather-plot'))

], style={'display': 'flex'}),

# Segment 2

html.Div([

html.Div(dcc.Graph(id='nas-plot')),

html.Div(dcc.Graph(id='security-plot'))

], style={'display': 'flex'}),

# Segment 3

html.Div(dcc.Graph(id='late-plot'), style={'width':'65%'})

])

""" Compute\_info function description

This function takes in airline data and selected year as an input and performs computation for creating charts and plots.

Arguments:

airline\_data: Input airline data.

entered\_year: Input year for which computation needs to be performed.

Returns:

Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.

"""

def compute\_info(airline\_data, entered\_year):

# Select data

df = airline\_data[airline\_data['Year']==int(entered\_year)]

# Compute delay averages

avg\_car = df.groupby(['Month','Reporting\_Airline'])['CarrierDelay'].mean().reset\_index()

avg\_weather = df.groupby(['Month','Reporting\_Airline'])['WeatherDelay'].mean().reset\_index()

avg\_NAS = df.groupby(['Month','Reporting\_Airline'])['NASDelay'].mean().reset\_index()

avg\_sec = df.groupby(['Month','Reporting\_Airline'])['SecurityDelay'].mean().reset\_index()

avg\_late = df.groupby(['Month','Reporting\_Airline'])['LateAircraftDelay'].mean().reset\_index()

return avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late

"""Callback Function

Function that returns fugures using the provided input year.

Arguments:

entered\_year: Input year provided by the user.

Returns:

List of figures computed using the provided helper function `compute\_info`.

"""

# Callback decorator

@app.callback( [

Output(component\_id='carrier-plot', component\_property='figure'),

Output(component\_id='weather-plot', component\_property='figure'),

Output(component\_id='nas-plot', component\_property='figure'),

Output(component\_id='security-plot', component\_property='figure'),

Output(component\_id='late-plot', component\_property='figure')

],

Input(component\_id='input-year', component\_property='value'))

# Computation to callback function and return graph

def get\_graph(entered\_year):

# Compute required information for creating graph from the data

avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_info(airline\_data, entered\_year)

# Line plot for carrier delay

carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrrier delay time (minutes) by airline')

# Line plot for weather delay

weather\_fig = px.line(avg\_weather, x='Month', y='WeatherDelay', color='Reporting\_Airline', title='Average weather delay time (minutes) by airline')

# Line plot for nas delay

nas\_fig = px.line(avg\_NAS, x='Month', y='NASDelay', color='Reporting\_Airline', title='Average NAS delay time (minutes) by airline')

# Line plot for security delay

sec\_fig = px.line(avg\_sec, x='Month', y='SecurityDelay', color='Reporting\_Airline', title='Average security delay time (minutes) by airline')

# Line plot for late aircraft delay

late\_fig = px.line(avg\_late, x='Month', y='LateAircraftDelay', color='Reporting\_Airline', title='Average late aircraft delay time (minutes) by airline')

return[carrier\_fig, weather\_fig, nas\_fig, sec\_fig, late\_fig]

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server()