# Create Dashboard using Plotly and Dash

Table of content



**Estimated time needed:** 45 minutes

# About Skills Network Cloud IDE

This Skills Network Labs Cloud IDE (Integrated Development Environment) provides a hands-on environment in your web browser for completing course and project related labs. It utilizes Theia, an open-source IDE platform, that can be run on desktop or on the cloud.
So far in the course you have been using Jupyter notebooks to run your python code. This IDE provides an alternative for editing and running your Python code. In this lab you will be using this alternative Python runtime to create and launch your Dash applications.

### Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. When you launch the Cloud IDE, you are presented with a 'dedicated computer on the cloud' exclusively for you. This is available to you as long as you are actively working on the labs.

Once you close your session or it is timed out due to inactivity, you are logged off, and this 'dedicated computer on the cloud' is deleted along with any files you may have created, dowloaded or installed. The next time you launch this lab, a new environment is created for you.

*If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan to your time accordingly and finish your labs in a single session.*

# Scenario:

The objective of this part of the Final Assignment is to analyze the historical trends in automobile sales during recession periods, as you did in the previous part. The goal is to provide insights into how the sales of XYZAutomotives, a company specializing in automotive sales, were affected during times of recession.

In this final assignment, you will have the opportunity to demonstrate the **Dashboarding** skills you have acquired in this course.

This lab aims to assess your abilities in creating various visualizations using *Plotly and Dash*. As a data scientist, you have been given a task to prepare a report on your finding from Automobile Sales data analysis.
You decided to develop a dashboard representing two main reports:-

- Yearly Automobile Sales Statistics' Recession Period Statistics

*NOTE:* Year range is between 1980 and 2013.

## Components of the report items

1. Yearly Automobile Sales Statistics
    - Yearly Average Automobile sales using line chart for the whole period.
    - For the chosen year provide,
        - Total Monthly Automobile sales using line chart.
        - Average Monthly Automobile sales of each vehicle type using bar chart.
        - Total Advertisement Expenditure for each vehicle using pie chart
2. Recession Period Statistics
    - Average Automobile sales using line chart for the Recession Period using line chart.
    - Average number of vehicles sold by vehicle type using bar chart
    - Total expenditure share by vehicle type during recession usssing pie chart
    - Effect of unemployment rate on vehicle type and sales using bar chart

*NOTE:* You have worked creating a dashboard components in `Flight Delay Time Statistics Dashboard` section. You will be working on the similar lines for this Dashboard

# Dataset Variables:

### Dataset Variables for your reference

The dataset includes the following variables

- *Date*: The date of the observation.
- *Recession*: A binary variable indicating recession perion; 1 means it was recession, 0 means it was normal.
- *Automobile_Sales*: The number of vehicles sold during the period.
- *GDP*: The per capita GDP value in USD.
- *Unemployment_Rate*: The monthly unemployment rate.
- *Consumer_Confidence*: A synthetic index representing consumer confidence, which can impact consumer spending and automobile purchases.
- *Seasonality_Weight*: The weight representing the seasonality effect on automobile sales during the period.
- *Price*: The average vehicle price during the period.
- Advertising_Expenditure: The advertising expenditure of the company.
- *Vehicle_Type*: The type of vehicles sold; Supperminicar, Smallfamiliycar, Mediumfamilycar, Executivecar, Sports.
- *Competition*: The measure of competition in the market, such as the number of competitors or market share of major manufacturers.
- *Month*: Month of the observation extracted from Date.
- *Year*: Year of the observation extracted from Date.

---

## Requirements to create the expected Dashboard

- Two dropdown menus: For choosing report type and year
- Each dropdown will be designed in a division

> The second dropdown (for selecting the year) should be **enabled** only if when the user selects "Yearly Statistics report" from the previous dropdown, else it should be disabled only. - The second dropdown (for selecting the year) should be **enabled** only if when the user selects "Yearly Statistics report" from the previous dropdown, else it should be disabled only.

- Layout for adding graphs.
- Callback functions to return to the layout and display graphs.
  - First callback will be required to take the input for the report type and set the years dropdown to be enabled to take the year input for "Years Statistics Report", else this dropdown be put on disabled.
  - In the second callback you will fetch the value of report type and year and return the required graphs appropriately for each type of report
- The four plots to be displayed in 2 rows, 2 column representation

NOTE:- For every task, you will required to save the screenshort/image, which you will be asked to submit for evaluation at the Final Submission stage.

# Get the application skeleton

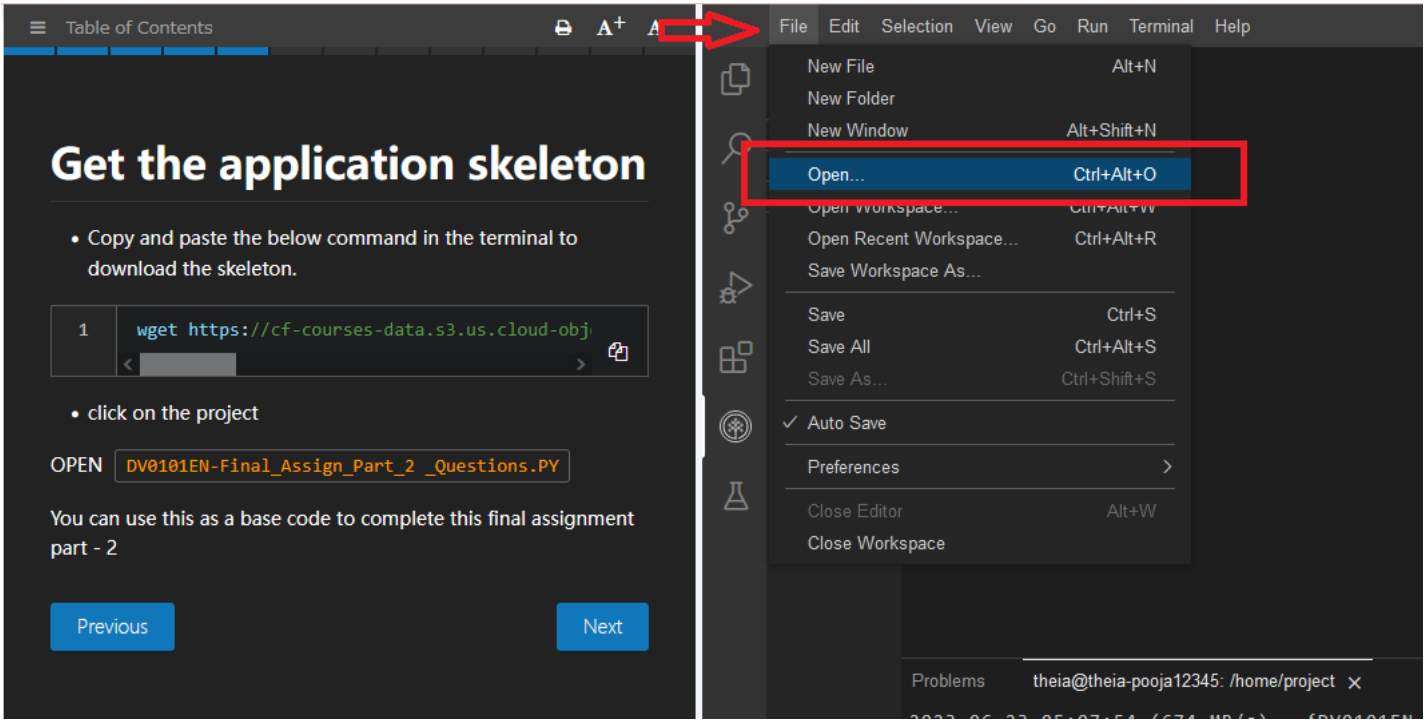
- Copy and paste the below command in the terminal to download the skeleton.

```
1. 1
```

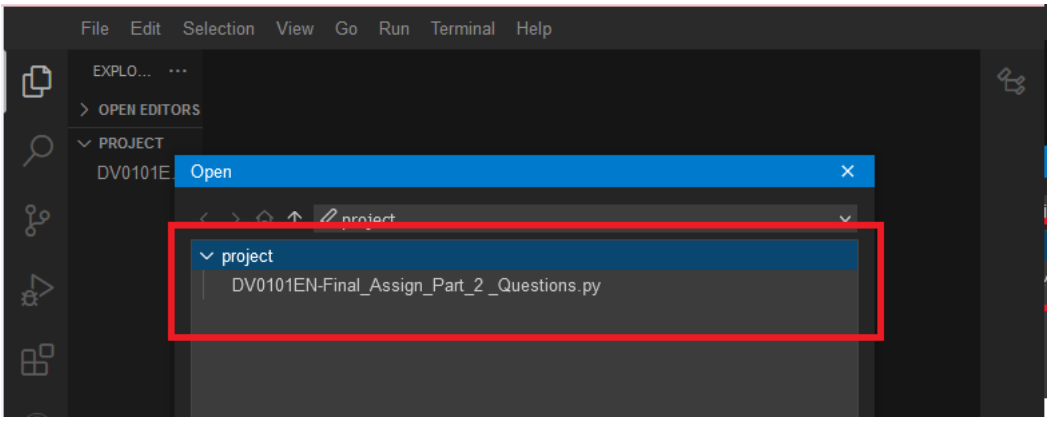
```
1. wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-DV0101EN-Coursera/labs/v4/Final_Project/DV0101EN
```

Copied!

- Click on File menu




OPEN `DV0101EN-Final_Assign_Part_2 _Questions.PY`




You can use this as a base code to complete this final assignment part 2

# Let's create the application

## Review

Search/Look for `Review` word in the script to learn how commands are used and computations are carried out.

## TASKS

Search/Look for `TASK` word in the script to identify places where you need to complete the code.

# TASK 2.1: Create a Dash application and give it a meaningful title

- Provide title of the dash application title as

  `Automobile Sales Statistics Dashboard`

- Make the heading center aligned
- set color as `#503D36`
- font size as `24`

  Sample: *style={'textAlign': 'left', 'color': '#000000', 'font-size': 0}*

### Reference link

NOTE: Once the application is up and running, take the screenshort representing the title of the application and save it as **'Title.png'**

---

# TASK 2.2: Add drop-down menus to your dashboard with appropriate titles and options

Create FIRST dropdown menu and add two *Report* options to it.
Below is the skeleton:

```
1
2
3
4
5
6
7
```

```
  dcc.Dropdown(id='....',
                    options=[
                            {'label': '....', 'value': '...'},
                            {'label': '....', 'value': '...'}
                            ],
                   placeholder='....',
                   style={....})
```

Copied!

Parameters to be updated in `dcc.Dropdown`:

- Set `id` as `'dropdown-statistics'`.
- Set `options` to list containing dictionaries with key as `label` and user provided value for labels in `value`.

  _1st option

  - label: Yearly Statistics
  - value: Yearly Statistics

  _2nd option

  - label: Recession Period Statistics
  - value: Recession Period Statistics
- Set placeholder to `Select a report type`.
- Set value to 'Select Statistics'
- OPTIONAL: Set width as `80%`, padding as `3px`, font size as `20px`, text-align-last as `center` inside style parameter dictionary.

---

Create SECOND dropdown menu for selecting the *YEAR* options to it

```
1
2
3
4
```

```
  dcc.Dropdown(id='....',
                    options=[{'label': i, 'value': i} for i in year_list],
                   placeholder='....',
                   style={....})
```

Copied!

- you can pass a list of years as options in this dropdowwn
  [hint: year_list = [i for i in range(1980, 2024, 1)]]
- Set id to 'select-year'

---

Reference link

# TASK 2.3: Add a division for output display with appropriate id and classname property

Add an inner division to display the output

```
1
2
3
```

```
html.Div([
```

```
    html.Div(id='........', className='..........', style={'display': 'flex'}),
])
```

Copied!

- Set id to ‘output-container’
- className to ‘chart-grid’
- style it to be displayed as a ‘flex’

For reference, observe how code under `REVIEW3` has been structured.

We will pass the plots as returned by the callback function into this output-container later refering to the class name of it.

---

NOTE: Take the screenshort representing the code snippet wherein you have created the output division and save it as **‘Outputdiv.png’**

---

# TASK 2.4: Creating Callbacks; Define the callback function to update the input container based on the selected statistics and the output container

We need two callbacks:-
1. to fetch the year, once ‘year statistic reports’ is selected
2. to create the output displays of the insights for respective report.

#### Fetching the year and disabling the input for other reportFetching the year and disabling the input for other report

Once the choice is made by the user, if it is yearly statistics, the input container shall get enabled else it will be in disabled state

This means the `output component` will be the `select year`,
while `input component` will be `dropdown-statistics`

and then we can make use of the if-else statement to return appropriate value to the component-property `disabled`. Refer to the below code snippet:

```
1
2
3
4
5
6
7
8
9
```

```
@app.callback(
    Output(component_id='....', component_property='disabled'),
    Input(component_id='....',component_property='....'))

def update_input_container(.....):
    if ..... =='Yearly Statistics':
        return False
    else:
        return True
```

Copied!

---

#### Callback for plotting

Our layout has 1 output container, and we will be required to return the plots developed `dcc.Graph()` into this container `as divisions`.
For each report we need to display four plots.
we will make use of returning division to the outer-container styled as flex for 2x2 display.

For this callback, there will be two input components : `select-year` and `dropdown-statistics`

While the output should be displayed in the `output-container`, as we opt to display the graphs with divisions using dcc.Graph(), we will make use of `children` property here

```
1
2
3
```

```
@app.callback(
    Output(component_id='....', component_property='children'),
    [Input(component_id='....', component_property='...'), Input(component_id='....', component_property='...')])
```

Copied!

You need to create seperate dataframe from the dataset according to the report type.
If the selected report is `'Recession Period Statistics'`, you can fetch the data by using conditions like below:-

```
1
2
3
4
5
```

```
def update_output_container(...., ...):
    if .... == 'Recession Period Statistics':
        # Filter the data for recession periods
        recession_data = data[data['Recession'] == 1]
        ......................
```

Copied!

On the same lines, you can create the dataframe, according to the year value entered by the user for `'Yearly Statistics report'`

---

NOTE: Take the screenshort representing the the code snippet wherein you have created the two callback functions and save it as **'Callbacks.png'**

---

# TASK5: 2.5 Creating Graphs

You will be required to prepare the data as per the plot requirement (usually using `groupby()`)

Then you need to Create a different plots using `plotly.express`.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
```

```
 # Plot 1 : Automobile sales fluctuate over Recession Period (year wise)
         # grouping data for plotting
        yearly_rec=recession_data.groupby('Year')['Automobile_Sales'].mean().reset_index()
        # Plotting the line graph
        R_chart1 = dcc.Graph(
            figure=px.line(yearly_rec,
                x='........',
                y='........',
                title="........"))
.........
## Plot 2:........
............
## Plot 3 : Pie chart for total expenditure share by vehicle type during recessions
            # grouping data for plotting
            exp_rec= recession_data.groupby(.............)
            R_chart3 = dcc.Graph(
                    figure=px.pie(.............,
                    values='............',
```

```
                names='.........',
                title="............"
               )
        )
.........
## Plot 4:........
............
 Note: Complete the TASK 2.6 for the Recession data before proceeding to Yearly data.

  # Yearly Statistic Report Plots
    elif (input_year and selected_statistics=='..............') :
        yearly_data = data[data['Year'] == ......]

.........
## Plot 1 :........
............
#Yearly Automobile sales using line chart for the whole period.
        yas= data.groupby('Year')['Automobile_Sales'].mean().reset_index()
        Y_chart1 = dcc.Graph(figure=px.line(................))

.........
## Plot 2 :........
............

 .........
## Plot 3 :........
............
  # Plot bar chart for average number of vehicles sold during the given year
        avr_vdata=yearly_data.groupby.......................
        Y_chart3 = dcc.Graph( figure................,title='Average Vehicles Sold by Vehicle Type in the year {}'.format(input_year)))

.........
## Plot 4 :........
............

```

Copied!

Link for reference:-

- Line Chart
- Bar Chart
- Pie Chart

NOTE: Label the plots properly.

---

NOTE: Once the application is up and running, take the screenshorts representing the graphs for each report type , [each graph should be clearly captures] and save it as **'Graphs.png'**

---

# TASK 2.6: Returning the graphs for display

We want the output to be displayed in a 2x2 grid like layout.

- we will return a list of two divisions to the output-container (className set as 'chart-grid'), each containing two divisions with style as 'flex'.

  [
  [div 1[[Plot 1], [Plot 2]] styled as flex,
  [div 2 [[Plot 3], [Plot 4]] styled as flex
  ]

Refer to the code snippet below:-

```
1
2
3
4
5
```

```
 return [
 ## Note: Value for the children parameter can be R_chart1, R_chart2 so on or Y_chart1,Y_chart2 so on depending on the type of charts.
            html.Div(className='chart-item', children=[html.Div(children=......),html.Div(children=......)],style={'display': 'flex'}),
            html.Div(className='chart-item', children=[html.Div(children=......),html.Div(children=......)],style={'display': '....'})
            ]
```

Copied!

NOTE: In the same way, return the four plots for Yearly report in the callback function.

---

NOTE: Once the application is up and running, take the screenshorts representing all four graphs for each report type and save it as **'Results.png'**

---

# Run the Application

- Firstly, install packaging, pandas and dash using the following command

```
1. 1
```

```
1. python3.8 -m pip install packaging
```

Copied!

```
1. 1
```

```
1. python3.8 -m pip install pandas dash
```

Copied!

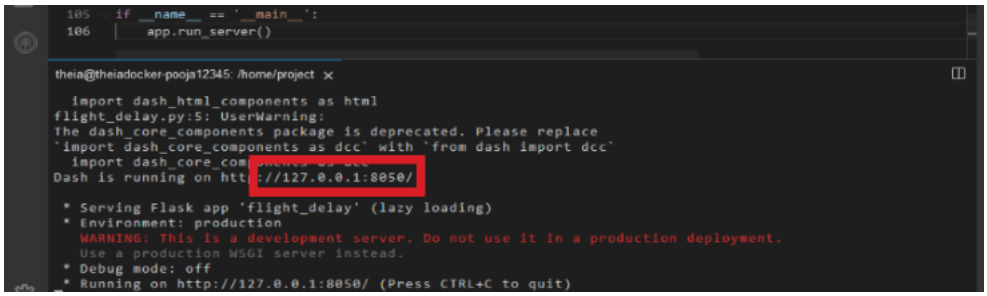
- Run the python file using the command

```
1. 1
```

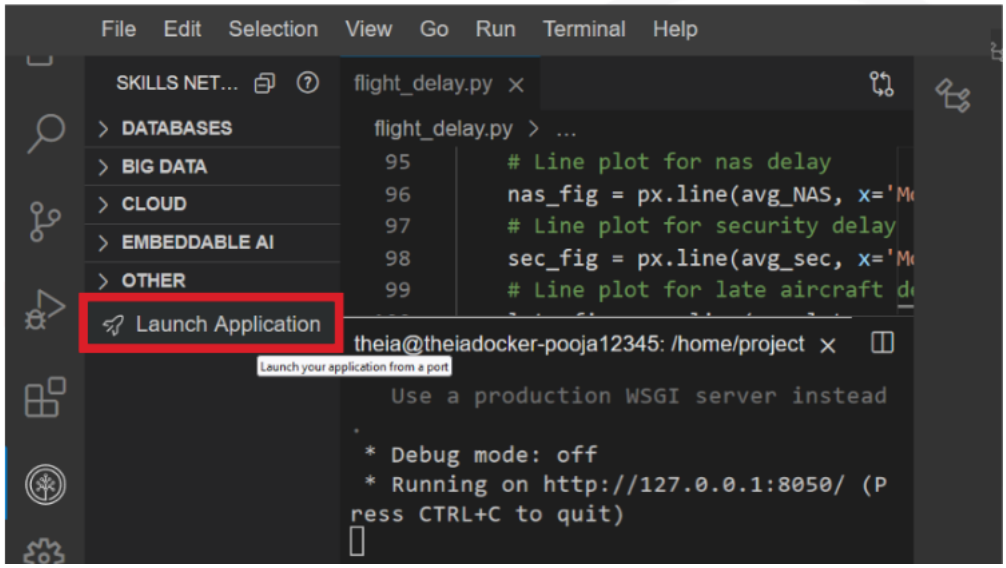
```
1. python3.8 DV0101EN-Final_Assign_Part_2_Questions.py
```

Copied!

- Observe the port number shown in the terminal.




- Click on the `Launch Application` option from the side menu bar. Provide the port number and click `OK`




**Congratulations, you have successfully completed your application!**

## Author

Dr. Pooja

## Changelog

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 2023-06-22 | 1.0 | Dr. Pooja | Initial Lab Creation |