**Solutions for Dash Fundamentals Assignment**

**Exercise A:** incorporate the dataset shades.csv into your app. And create the following layout in one app file:

1. A [Dropdown](https://dash.plotly.com/dash-core-components/dropdown) that uses the column brand as the dropdown options. Make sure the brand names are unique (do not repeat themselves). Then, assign “Revlon” as the initial value.

*from* dash *import* Dash, dcc, html

*import* pandas *as* pd

df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv')

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

app.layout = html.Div([

dcc.Dropdown(options=df.brand.unique(), value="Revlon")

])

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

app.run(debug=*True*)



2. A [RadioItems](https://dash.plotly.com/dash-core-components/radioitems) component in which the values from the column named group are assigned to the options property. The options should be unique and sorted from 0 to 7.

*from* dash *import* Dash, dcc, html

*import* pandas *as* pd

df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv')

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

app.layout = html.Div([

dcc.Dropdown(options=df.brand.unique(), value="Revlon"),

dcc.RadioItems(options=sorted(df.group.unique()))

])

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

app.run(debug=*True*)



3. Update the options property of the [RadioItems](https://dash.plotly.com/dash-core-components/radioitems) component so that the values (of the options) represent numbers from 0 to 7, but the labels are their respective strings ([see Readme-shades](https://github.com/plotly/datasets/blob/master/Dash-Course/makeup-shades/README-shades.md) for the strings).

*from* dash *import* Dash, dcc, html

*import* pandas *as* pd

df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv')

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

app.layout = html.Div([

dcc.Dropdown(options=df.brand.unique(), value="Revlon"),

dcc.RadioItems(options=[{"label": "Fenty Beauty's PRO FILT'R Foundation Only", "value": 0},

{"label": "Make Up For Ever's Ultra HD Foundation Only", "value": 1},

{"label": "US Best Sellers", "value": 2},

{"label": "BIPOC-recommended Brands with BIPOC Founders", "value": 3},

{"label": "BIPOC-recommended Brands with White Founders", "value": 4},

{"label": "Nigerian Best Sellers", "value": 5},

{"label": "Japanese Best Sellers", "value": 6},

{"label": "Indian Best Sellers", "value": 7}])

])

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

app.run(debug=*True*)



**Exercise B:** using the same shades.csv create another app that incorporates Dash AG Grid into the layout:

1. The [Dash AG Grid](https://dash.plotly.com/dash-ag-grid/getting-started) should represent the complete dataset with all its columns.

import micropip

await micropip.install("dash\_ag\_grid")  
from dash import Dash, dcc, html

import dash\_ag\_grid as dag

import pandas as pd

df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv')

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

grid = dag.AgGrid(

id="my-table",

rowData=df.to\_dict("records"),

columnDefs=[{"field": i} for i in df.columns]

)

app.layout = html.Div([grid])

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)



2. Using [Pagination](https://dash.plotly.com/dash-ag-grid/pagination), add automatic pagination to Dash AG Grid and make sure all columns fit into the screen with no horizontal scroll bar (using the columnSize property).

import micropip

await micropip.install("dash\_ag\_grid")

from dash import Dash, dcc, html

import dash\_ag\_grid as dag

import pandas as pd

df = pd.read\_csv('<https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv>')

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

grid = dag.AgGrid(

id="my-table",

rowData=df.to\_dict("records"),

columnDefs=[{"field": i} for i in df.columns],

columnSize="sizeToFit",

dashGridOptions={"pagination": True, "paginationAutoPageSize": True},

)

app.layout = html.Div([grid])

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)



**Exercise C:** using the same shades.csv create a new app, where the layout has two new [Dash Core Components](https://dash.plotly.com/dash-core-components) that you haven’t used so far.

There is no solution to exercise C. The goal is to choose whichever components you prefer to practice with.

**Exercise D:** using the following [scatter plot example](https://plotly.com/python/line-and-scatter/#setting-size-and-color-with-column-names), add a scatter plot to your app that displays V (value/brightness) on the x-axis and S (saturation) on the y-axis.

Clue: to display the plot in the layout, remember to assign your plot to the figure property of the dcc.Graph, for example: dcc.Graph(figure=my\_scatter\_plot)

from dash import Dash, dcc, html

import plotly.express as px

import pandas as pd

df = pd.read\_csv('<https://raw.githubusercontent.com/plotly/datasets/master/Dash-Course/makeup-shades/shades.csv>')

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

fig = px.scatter(df, x='V', y='S')

app.layout = html.Div([dcc.Graph(figure=fig)])

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

