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 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.

```python
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 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.

```python
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 component so that the `values` (of the options) represent numbers from 0 to 7, but the `labels` are their respective strings (see Readme-shades for the strings).

Python

```
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 should represent the complete dataset with all its columns.

Python

```
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-Cour
se/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, 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).

Python

```
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-s
hades/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 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, 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)`

Python

```
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)
```