# 99 - Solution - Diabetes Analysis Dashboard

April 15, 2023

[ ]:
```
!pip install jupyter-dash
```

[ ]:
```
# Uses a dropdown list to control the year of the chart
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

###### Cleaning up the data after it is read in.  Fixing values, dropping␣
 ↪columns, creating 2 aggregate tables.

diabetes = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪b2aa6df015816ec35afc482b53df1b7ca7a31f80/diabetes_for_plotly.csv')
diabetes['gender'] = diabetes['gender'].replace({'M':'Male', 'Mle':'Male', 'F':
 ↪'Female',
                                                 'female':'Female', 'male':
 ↪'Male',
                                                 '?':'Female', 'Unknown/
 ↪Invalid':'Female'})
diabetes = diabetes.drop('encounter_id',axis=1)
diabetes = diabetes.drop('patient_nbr',axis=1)
diabetes = diabetes.drop('admission_type_id',axis=1)
diabetes = diabetes.drop('discharge_disposition_id',axis=1)
diabetes = diabetes.drop('diag_1',axis=1)
diabetes = diabetes.drop('A1Cresult',axis=1)
diabetes = diabetes.drop('insulin',axis=1)
diabetes = diabetes.drop('diabetesMed',axis=1)
diabetes = diabetes.drop('readmitted',axis=1)
d_month = diabetes.
 ↪groupby('month')[['time_in_hospital','num_lab_procedures','num_procedures','num_medications
 ↪sum().reset_index()
```

```
d_gender = diabetes.
 ↪groupby('gender')[['time_in_hospital','num_lab_procedures','num_procedures','num_medication
 ↪sum().reset_index()
```

# 1 Exercise 1 - plotly charts - 30 minutes

- read in the diabetes_for_plotly dataset (already done above)
- group data as needed
- Use express or graph objects
- Create a scatter plot of any two measures. Use a third measure to adjust the size. Color by a categorical value. Add hover text to show the age group.
- Create a side-by-side bar chart showing number of lab procedures and number of non lab procedures by gender.
- Create a line chart showing number of number of medications by month.
- Create a line chart showing number of number of procedures by month.
- Create a fifth chart of your choice (NOT scatter, bar or line) using the documentation.

### 1.0.1 scatterplot is given a variable name: labs

[ ]:
```
# express version
# Create a scatter plot of any two measures. Use a third measure to adjust the␣
 ↪size. Color by a categorical value.
# Add hover text to show the age group.
labs = px.scatter(diabetes, x=diabetes.num_lab_procedures,
                  y=diabetes.num_medications,
                 # size = diabetes.time_in_hospital,
                 # color = diabetes.gender,
                 # hover_data = ['age']
                  )
labs.show()
```

[ ]:
```
# go version
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=diabetes.num_lab_procedures,
    y=diabetes.num_medications,
    mode = 'markers',
    #marker_color='indianred'
    marker_color = diabetes.time_in_hospital
))

fig.show()
```

[ ]:
```
# express version
# Create a side-by-side bar chart showing number of lab procedures and number␣
 ↪of non lab procedures by gender.
```

```
fig = px.bar(d_gender, x='gender', y=['num_lab_procedures', 'num_procedures'],␣
 ↪barmode = 'group')
fig.show()
```

[ ]:
```
# go version
fig = go.Figure(
    data=[go.Bar(name = 'labs', x=d_gender.gender, y = d_gender.
 ↪num_lab_procedures),
          go.Bar(name = 'non labs', x=d_gender.gender, y = d_gender.
 ↪num_procedures)],
    layout=go.Layout(
        title=go.layout.Title(text="A Figure Specified By A Graph Object")
    )
)
fig.show()
```

[ ]:
```
# Create a line chart showing number of number of medications by monmth.

fig = px.line(d_month,x='month', y='num_medications')
fig.show()

# NOTE:
#    fig = go.Figure(go.Scatter(x=d_month.month, y=d_month.
 ↪num_medications,mode='lines')) DEFAULT is a line
```

[ ]:
```
# Create a line chart showing number of number of procedures by month.

fig = px.line(d_month,x='month', y='num_procedures')
fig.show()
```

[ ]:
```
# Create a line chart showing number of number of procedures by gender.

fig = px.bar(d_gender,x='gender', y='num_procedures')
fig.show()
```

# 2 Dash Exercise 2 - 20 minutes

## 2.1 Create a dashboard with one chart

- Add a new cell that will contain all of the dashboard code.
- Add dashboard code to show any chart created earlier.

[ ]:
```
app = JupyterDash(__name__)

procs = px.bar(d_gender, x='gender', y=['num_lab_procedures',␣
 ↪'num_procedures'], barmode = 'group')
```

```
# Add dcc.Graph code here
app.layout = html.Div([

    html.Div([dcc.Graph(id='x', figure=procs )]),

])

app.run_server(mode='inline')
#app.run_server(mode='external', port = 8202)
```

# 3 Dash Exercise 3 - 15 minutes

## 3.1 Add two more charts to the dashboard

- Copy the code from the prior exercise and add to it.
- Add two more of your graphs to the dashboard
- Place each chart in its own Div
- Give each chart a different background and text color
- Try some of your own formatting. Use the documentation.

[ ]:
```
app = JupyterDash(__name__)

procs = px.bar(d_gender, x='gender', y=['num_lab_procedures',␣
 ↪'num_procedures'], barmode = 'group')
fig2 = px.bar(d_gender,x='gender', y='num_procedures')
fig3 = px.line(d_month,x='month', y='num_procedures')

procs.update_layout(plot_bgcolor='#111111')
fig2.update_layout(plot_bgcolor='red')
fig3.update_layout(plot_bgcolor='green')

colors = {
    'background': 'purple', # black
    'text': '#7FDBFF'         # light blue
}


app.layout = html.Div([

    html.Div([dcc.Graph(id='x', figure=procs )], style={'width':␣
 ↪'49%','textAlign': 'left','color': colors['text']}),
    html.Div([dcc.Graph(id='y', figure=fig2 )]),
    html.Div([dcc.Graph(id='z', figure=fig3 )])

])
```

```
app.run_server(mode='inline')
#app.run_server(mode='external', port = 8203)
```

# 4 Dash Exercise 4 - 15 minutes

## 4.1 Add a core component that uses a callback

- Add a checkbox core component to the dashboard
- Use the checkbox to select one or more gender values
- Set the default to select all genders
- Apply the selection to all of the charts

### 4.1.1 Adding callback feature

[ ]:
```
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

app = JupyterDash(__name__)

diabetes = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪b2aa6df015816ec35afc482b53df1b7ca7a31f80/diabetes_for_plotly.csv')
diabetes['gender'] = diabetes['gender'].replace({'M':'Male', 'Mle':'Male', 'F':
 ↪'Female',
                                                 'female':'Female', 'male':
 ↪'Male',
                                                 '?':'Female', 'Unknown/
 ↪Invalid':'Female'})
d_month = diabetes.
 ↪groupby('month')[['time_in_hospital','num_lab_procedures','num_procedures','num_medications
 ↪sum().reset_index()
d_gender = diabetes.
 ↪groupby('gender')[['time_in_hospital','num_lab_procedures','num_procedures','num_medication
 ↪sum().reset_index()

app.layout = html.Div([

    dcc.Checklist(
        id = 'checklist',
        options = ['Female', 'Male'],
```

```
        value = ['Female']),

    html.Div([dcc.Graph(id='graph')])
])

@app.callback(
    Output('graph', 'figure'),
    Input('checklist', 'value'))

def update_charts(abc):

    df = d_gender[d_gender['gender'].isin(abc)]
    procs = px.bar(df, x='gender', y=['num_lab_procedures', 'num_procedures'],␣
 ↪barmode = 'group')
    return procs


app.run_server(mode='inline')
#app.run_server(mode='external', port = 8204)
```

## 5 Exercise 5 - Change the scatterplot variables - 30 minutes

- modify your Diabetes Dashboard.
- Use the code above (as an example) to have dropdown list that change the data in the scatter plot.

[ ]:
```
# Changing the variables to include in the plot

app = JupyterDash(__name__)

diabetes = pd.read_csv('https://bitbucket.org/jimcody/sampledata/raw/
 ↪b2aa6df015816ec35afc482b53df1b7ca7a31f80/diabetes_for_plotly.csv')
diabetes['gender'] = diabetes['gender'].replace({'M':'Male', 'Mle':'Male', 'F':
 ↪'Female',
                                                 'female':'Female', 'male':
 ↪'Male',
                                                 '?':'Female', 'Unknown/
 ↪Invalid':'Female'})

######## NOTICE: d_month is now grouped by year and month

d_month = diabetes.
 ↪groupby(['year','month'])[['time_in_hospital','num_lab_procedures','num_procedures','num_me
 ↪sum().reset_index()
```

```
d_gender = diabetes.
 ↪groupby('gender')[['time_in_hospital','num_lab_procedures','num_procedures','num_medication
 ↪sum().reset_index()


available_variables = d_month.select_dtypes(exclude = ['object'])
available_variables = available_variables.drop('month',axis=1)
available_variables = available_variables.drop('year',axis=1)
#available_variables

app.layout = html.Div([

    html.Div([
        html.Div([
            dcc.Dropdown(
                id='xaxis-column',
                options=[{'label': i, 'value': i} for i in available_variables],
                value='num_procedures')
        ],style={'width': '48%', 'display': 'inline-block'}),
        html.Div([
            dcc.Dropdown(
                id='yaxis-column',
                options=[{'label': i, 'value': i} for i in available_variables],
                value='num_lab_procedures')
        ], style={'width': '48%', 'float': 'right', 'display': 'inline-block'}),


    dcc.Graph(id='fig1'),
    'Select a Year:  ',
    html.P(),
    dcc.RadioItems(id='year-s', options=[2019,2020,2021], value=2019)  ])
    ])

@app.callback(
    Output('fig1','figure'),
    Input('year-s','value'),
    Input('xaxis-column','value'),
    Input('yaxis-column','value'))

def update_figure(year, xa, ya):
    filtered_df = d_month[d_month.year == year]

    labs = px.scatter(filtered_df,
                      x= xa,
                      y= ya)

    labs.update_layout(transition_duration=500)
```

```
    return labs


app.run_server(mode='inline')
#app.run_server(mode='external', port = 8205)
```

[ ]: