***HOW TO RUN:***

Dashboard

Yifei Ran–ranyy008

Dashboard: Code documentation

1. A sensor\_data.csv is required: it should have the following table headers and appropriate data types.

*[time,sensor\_one\_temp,sensor\_one\_warning,sensor\_one\_alarm,sensor\_one\_emergency,sensor\_one\_vib,sensor\_one\_warning\_vib,sensor\_one\_alarm\_vib,sensor\_one\_emergency\_vib,sensor\_two\_temp,sensor\_two\_warning,sensor\_two\_alarm,sensor\_two\_emergency,sensor\_two\_temp\_vib,sensor\_two\_warning\_vib,sensor\_two\_alarm\_vib,sensor\_two\_emergency\_vib,sensor\_three\_temp,sensor\_three\_warning,sensor\_three\_alarm,sensor\_three\_emergency,sensor\_three\_temp\_vib,sensor\_three\_warning\_vib,sensor\_three\_alarm\_vib,sensor\_three\_emergency\_vib]*

1. Run code: open Dashboard.py in PyCharm and run
2. 
3. Sensor\_Data.csv should appear in the folder
4. 

***Code explanation:***

1. **Imports**

import dash  
from dash.dependencies import Input, Output  
import dash\_table  
import dash\_core\_components as dcc  
import dash\_html\_components as html  
import pandas as pd

dash: is used to data analysis and visualization

Input, Output: is used to input and output

dash\_table: is used to make dash table

dash\_core\_components: is used to make web-based interactive components

dash\_html\_components: is used to make html components

pandas: is used to get and manipulate data in csv files

1. **Functions**
   1. Open the csv file generated in the DES section, because some columns are not number, that can not used in the bar chart and line graph, and if put them in the data table, the data table will be many gaps, so I drop some columns.



This is what the datatable would look like if the columns were not dropped, you can see that there are a lot of empty spaces and it's not very pretty.

df = pd.read\_csv('sensor\_data.csv')  
# open the csv file to get data  
df.drop(['sensor\_one\_warning','sensor\_one\_alarm','sensor\_one\_emergency','sensor\_one\_warning\_vib',  
 'sensor\_one\_alarm\_vib','sensor\_one\_emergency\_vib','sensor\_two\_warning','sensor\_two\_alarm',  
 'sensor\_two\_emergency','sensor\_two\_warning\_vib','sensor\_two\_alarm\_vib','sensor\_two\_emergency\_vib',  
 'sensor\_three\_warning', 'sensor\_three\_alarm', 'sensor\_three\_emergency', 'sensor\_three\_warning\_vib',  
 'sensor\_three\_alarm\_vib', 'sensor\_three\_emergency\_vib'],axis=1,inplace=True)

* 1. Make a title ‘Dashboard’ for html

app.layout = html.Div([  
html.Div([  
 html.H3('Dashboard')  
]),

* 1. Define the detail of the datable, include editing the cells, all data is passed to the table up-front or not ('none'), number of rows visible per page, filtering by column, align text columns to left. By default they are aligned to right.

Because the title of some columns are so long, so I make the width big.

dash\_table.DataTable(  
 id='datatable-advanced-filtering',  
 columns=[  
 {'name': i, 'id': i, 'deletable': True} for i in df.columns  
 # omit the id column  
 if i != 'id'  
 ],  
 data=df.to\_dict('records'),  
 editable=True, #editing the cells  
 page\_action='native', # all data is passed to the table up-front or not ('none')  
 page\_size=10, # number of rows visible per page  
 filter\_action="native", #filtering by column  
 style\_cell={  
 'textAlign':'left', # align text columns to left. By default they are aligned to right  
 'minWidth': 210,'maxWidth': 210,'Width': 210,  
 # couse some of the texts are too long so I make the width big  
 'backgroundColor':'white'  
 },  
),

* 1. Bar chart: use callback to get output and input.

@app.callback(  
 Output('bar-chart', 'children'),  
 Input('datatable-advanced-filtering', "derived\_virtual\_data"),  
)

* 1. Get all rows data from sensor\_data.csv

def update\_bar(all\_rows\_data):  
 dff = pd.DataFrame(all\_rows\_data) #get data from the csv file

* 1. Set the id of bar chart, and use the data in the csv file to make bar chart, because we have six columns’ different data, so I set the data for ‘x’ , ‘y’, ‘type’ and ‘name’ for each bar.

dcc.Graph(  
 id='bar-chart',  
 figure={  
 'data':[  
 # six bars for six different data  
 {'x': df['time'], 'y': df['sensor\_one\_temp'], 'type':'bar', 'name':'sensor\_one\_temp'},  
 {'x': df['time'], 'y': df['sensor\_one\_vib'], 'type':'bar', 'name':'sensor\_one\_vib'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp'], 'type':'bar', 'name': 'sensor\_two\_temp'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp\_vib'], 'type':'bar', 'name': 'sensor\_two\_temp\_vib'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp'], 'type':'bar', 'name': 'sensor\_three\_temp'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp\_vib'], 'type':'bar', 'name': 'sensor\_three\_temp\_vib'}  
 ],

* 1. Set the title for barchart

'layout':{  
 'title': 'Bar Chart' #the title of the bar chart  
}

* 1. Be similier with bar chart, for line graph we also start from use callback to get output and input.

@app.callback(  
 Output('line-graph', 'children'),  
 Input('datatable-advanced-filtering', "derived\_virtual\_data"),  
)

* 1. Get all rows data from sensor\_data.csv

def update\_line(all\_rows\_data):  
 dff = pd.DataFrame(all\_rows\_data)

* 1. Make six lines for six different data.

dcc.Graph(  
 id='line-graph',  
 figure={  
 'data':[  
 # six lines for six different data  
 {'x': df['time'], 'y': df['sensor\_one\_temp'], 'type':'line', 'name':'sensor\_one\_temp'},  
 {'x': df['time'], 'y': df['sensor\_one\_vib'], 'type':'line', 'name':'sensor\_one\_vib'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp'], 'type':'line', 'name': 'sensor\_two\_temp'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp\_vib'], 'type':'line', 'name': 'sensor\_two\_temp\_vib'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp'], 'type':'line', 'name': 'sensor\_three\_temp'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp\_vib'], 'type':'line', 'name': 'sensor\_three\_temp\_vib'}  
 ],

* 1. Set the title for line graph

'layout':{  
 'title': 'Line Graph' #the title of the line Graph

1. **Full code**

import dash  
from dash.dependencies import Input, Output  
import dash\_table  
import dash\_core\_components as dcc  
import dash\_html\_components as html  
import pandas as pd  
  
df = pd.read\_csv('sensor\_data.csv')  
# open the csv file to get data  
df.drop(['sensor\_one\_warning','sensor\_one\_alarm','sensor\_one\_emergency','sensor\_one\_warning\_vib',  
 'sensor\_one\_alarm\_vib','sensor\_one\_emergency\_vib','sensor\_two\_warning','sensor\_two\_alarm',  
 'sensor\_two\_emergency','sensor\_two\_warning\_vib','sensor\_two\_alarm\_vib','sensor\_two\_emergency\_vib',  
 'sensor\_three\_warning', 'sensor\_three\_alarm', 'sensor\_three\_emergency', 'sensor\_three\_warning\_vib',  
 'sensor\_three\_alarm\_vib', 'sensor\_three\_emergency\_vib'],axis=1,inplace=True)  
#delete the unuseful columns  
  
  
app = dash.Dash(\_\_name\_\_)  
  
app.layout = html.Div([  
html.Div([  
 html.H3('Dashboard')  
]),  
# make a title for dashboard  
  
html.Div([  
  
 html.Br(),  
 #define some id, these will be used later  
 html.Div(id='bar-chart'),  
 html.Div(id='line-graph'),  
  
# define the detail of datatable  
 dash\_table.DataTable(  
 id='datatable-advanced-filtering',  
 columns=[  
 {'name': i, 'id': i, 'deletable': True} for i in df.columns  
 # omit the id column  
 if i != 'id'  
 ],  
 data=df.to\_dict('records'),  
 editable=True, #editing the cells  
 page\_action='native', # all data is passed to the table up-front or not ('none')  
 page\_size=10, # number of rows visible per page  
 filter\_action="native",# filtering by column  
 style\_cell={  
 'textAlign':'left', # align text columns to left. By default they are aligned to right  
 'minWidth': 210,'maxWidth': 210,'Width': 210,  
 # couse some of the texts are too long so I make the width big  
 'backgroundColor':'white'  
 },  
 ),  
 html.Hr(),  
 html.Div(id='datatable-query-structure', style={'whitespace': 'pre'})  
  
])  
])  
  
  
#bar chart  
@app.callback(  
 Output('bar-chart', 'children'),  
 Input('datatable-advanced-filtering', "derived\_virtual\_data"),  
)  
def update\_bar(all\_rows\_data):  
 dff = pd.DataFrame(all\_rows\_data) #get data from the csv file  
 if "time" in dff and "sensor\_one\_temp" in dff:  
 return [  
 dcc.Graph(  
 id='bar-chart',  
 figure={  
 'data':[  
 # six bars for six different data  
 {'x': df['time'], 'y': df['sensor\_one\_temp'], 'type':'bar', 'name':'sensor\_one\_temp'},  
 {'x': df['time'], 'y': df['sensor\_one\_vib'], 'type':'bar', 'name':'sensor\_one\_vib'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp'], 'type':'bar', 'name': 'sensor\_two\_temp'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp\_vib'], 'type':'bar', 'name': 'sensor\_two\_temp\_vib'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp'], 'type':'bar', 'name': 'sensor\_three\_temp'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp\_vib'], 'type':'bar', 'name': 'sensor\_three\_temp\_vib'}  
 ],  
 'layout':{  
 'title': 'Bar Chart' #the title of the bar chart  
 }  
 }  
 )  
 ]  
  
# line graph  
@app.callback(  
 Output('line-graph', 'children'),  
 Input('datatable-advanced-filtering', "derived\_virtual\_data"),  
)  
def update\_line(all\_rows\_data):  
 dff = pd.DataFrame(all\_rows\_data)  
 if "time" in dff and "sensor\_one\_temp" in dff:  
 return [  
 dcc.Graph(  
 id='line-graph',  
 figure={  
 'data':[  
 # six lines for six different data  
 {'x': df['time'], 'y': df['sensor\_one\_temp'], 'type':'line', 'name':'sensor\_one\_temp'},  
 {'x': df['time'], 'y': df['sensor\_one\_vib'], 'type':'line', 'name':'sensor\_one\_vib'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp'], 'type':'line', 'name': 'sensor\_two\_temp'},  
 {'x': df['time'], 'y': df['sensor\_two\_temp\_vib'], 'type':'line', 'name': 'sensor\_two\_temp\_vib'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp'], 'type':'line', 'name': 'sensor\_three\_temp'},  
 {'x': df['time'], 'y': df['sensor\_three\_temp\_vib'], 'type':'line', 'name': 'sensor\_three\_temp\_vib'}  
 ],  
 'layout':{  
 'title': 'Line Graph' #the title of the line Graph  
 }  
 }  
 )  
 ]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server(debug=True)