Name: **Swarup Tripathy** Assignment Number: **3**
Course: **ECE3502: IoT Domain Analyst** Date: March 6, 2022

Reg No: 19BEE0167

1 Problem 1

Create the interactive dashboard using python. Use CSV file Vgsales to interactively display various charts like bar, pie etc to illustrate the interactive display. Implement this for covid-19 cases using your own CSV file.

2 Python Code

```
# Importing all the necessary libraries
import dash
import pandas as pd
import plotly.express as px

from dash import html
from dash import dcc
from dash.dependencies import Input, Output

df=pd.read_csv("C:/Users/iwill/Desktop/VIT/Semester 6/IoT Lab/Lab 5/vgsales.csv")
print(df[:5])  # to display first 5 data
```

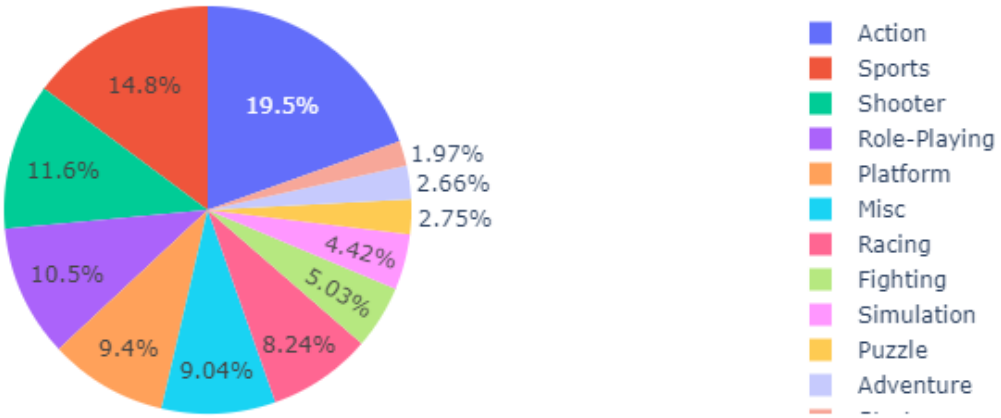
2.1 Code output

```
    Rank             Name Platform  Year     Genre   Publisher  NA_Sales  \
0   259         Asteroids     2600  1980   Shooter       Atari      4.00
1   545  Missile Command      2600  1980   Shooter       Atari      2.56
2  1768          Kaboom!     2600  1980      Misc  Activision      1.07
3  1971         Defender     2600  1980      Misc       Atari      0.99
4  2671           Boxing     2600  1980  Fighting  Activision      0.72

   EU_Sales  JP_Sales  Other_Sales  Global_Sales
0      0.26       0.0         0.05          4.31
1      0.17       0.0         0.03          2.76
2      0.07       0.0         0.01          1.15
3      0.05       0.0         0.01          1.05
4      0.04       0.0         0.01          0.77
```

3 Python Code

```
fig_pie=px.pie(data_frame=df, names="Genre", values="Global_Sales")
fig_pie.show()
```

3.1 Code output



4 Python Code

```
print(sorted(df.Year.unique()))
print(df.Genre.nunique())
print(df.Genre.unique())
print(len(df.Genre.unique()))
```

4.1 Code output

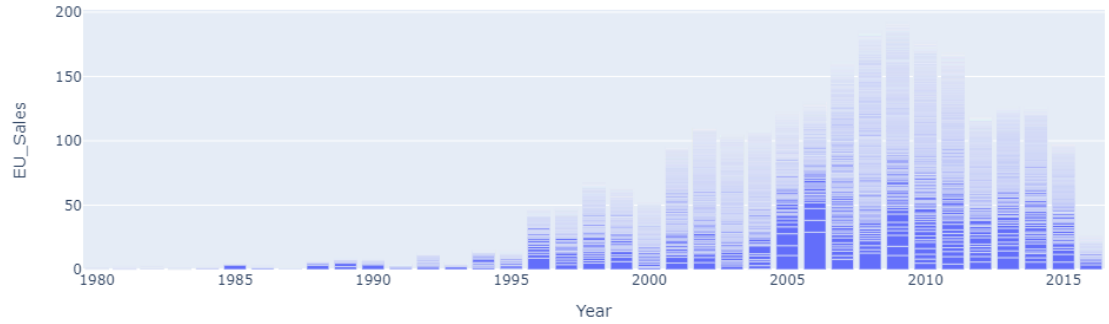
```
[1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994,
1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009,
2010, 2011, 2012, 2013, 2014, 2015, 2016]
12
['Shooter' 'Misc' 'Fighting' 'Sports' 'Action' 'Platform' 'Puzzle'
 'Racing' 'Simulation' 'Adventure' 'Role-Playing' 'Strategy']
12
```

5 Python Code

```
print(df.columns)   #To check the column headings

fig_bar = px.bar(data_frame=df,x="Year", y="EU_Sales")
fig_bar.show()
```

5.1 Code output

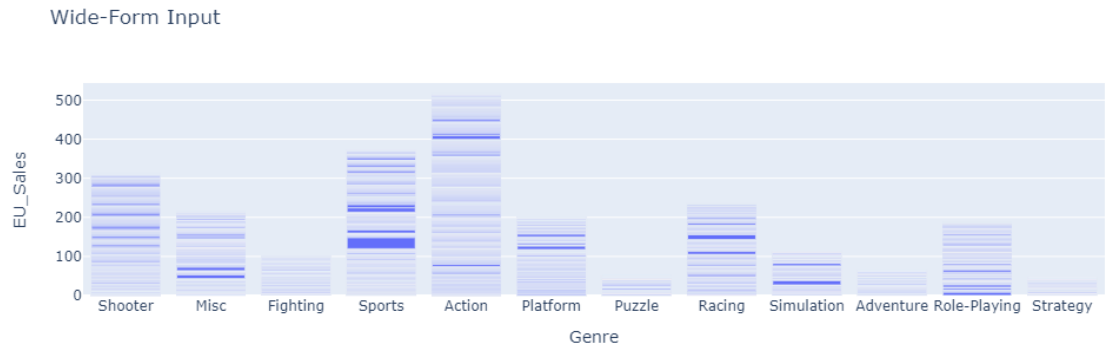
```
Index(['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher', 'NA_Sales',
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
```



6 Python Code

```
fig_bar = px.bar(data_frame=df,x="Genre", y="EU_Sales",title="Wide-Form Input")
fig_bar.show()
```

6.1 Code output



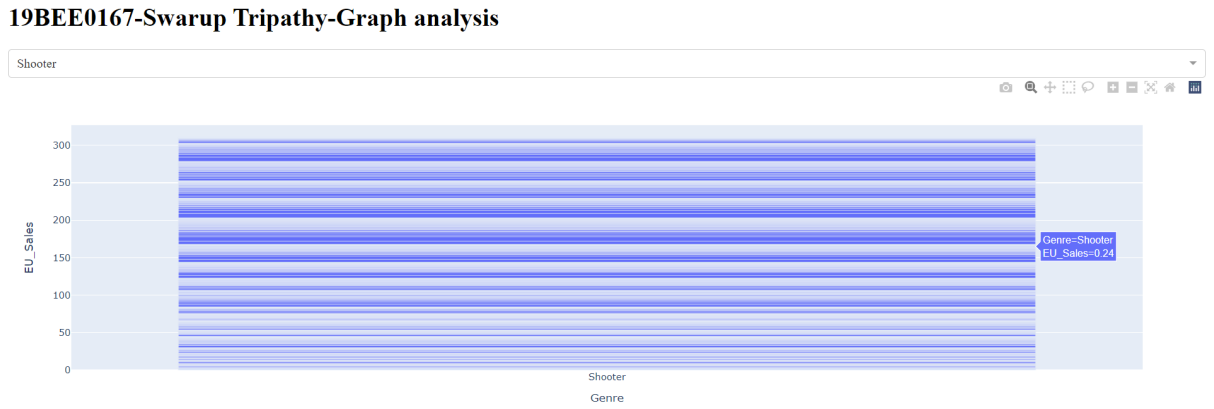
7 Final Python Code on DASH application

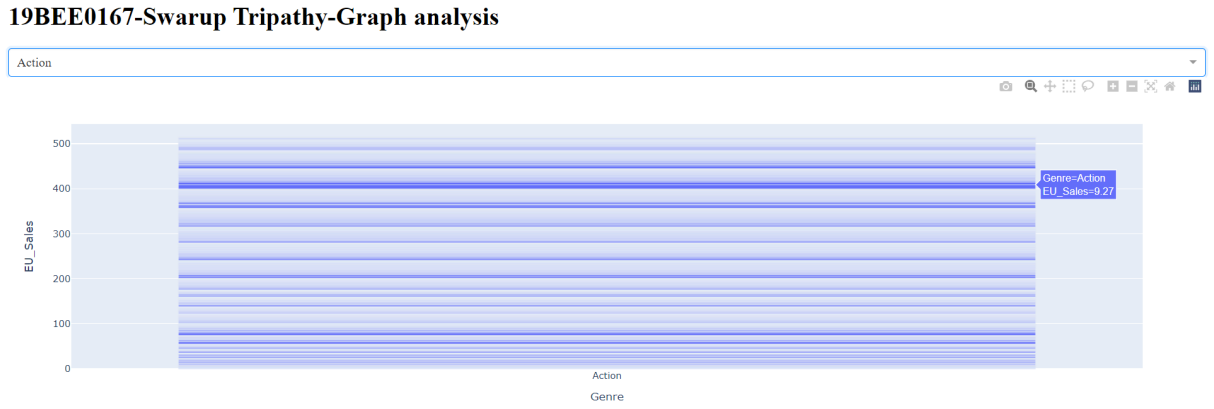
```
##############################################################################
################ GRAPHS BEING DISPLAYED FOR EVERY GENRE ######################
##############################################################################

import dash
import plotly.express as px
import pandas as pd
from dash import html
from dash import dcc
from dash.dependencies import Input, Output

# import dash_core_components as dcc
# import dash_html_components as html

df = pd.read_csv("C:/Users/iwill/Desktop/VIT/Semester 6/IoT Lab/Lab 5/vgsales.csv")
```

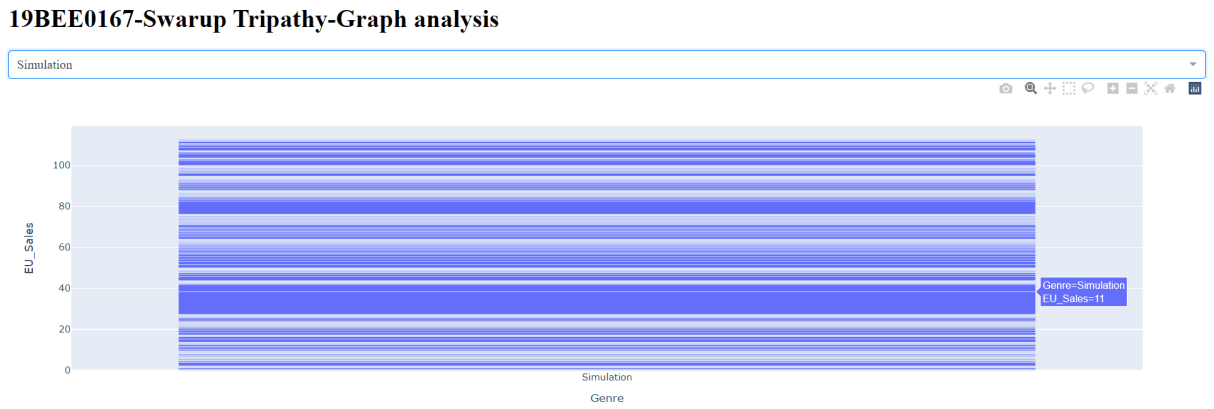
```
# print(df[:5])
# print(df.loc[:5,["Global_Sales"]])

gen = df.Genre.unique()

app = dash.Dash(__name__)
app.layout=html.Div([
    html.H1("19BEE0167-Swarup Tripathy-Graph analysis"),
    dcc.Dropdown(
        id='Genre-choice',
        options =[{'label':x,'value':x} for x in gen],
        value=gen[0],
        clearable=False),
    dcc.Graph(id="bar-chart")
])

@app.callback(
    Output(component_id="bar-chart", component_property='figure'),
    Input(component_id='Genre-choice', component_property='value')
)

def interactive_graphing(Genre):
    mask = df["Genre"] == Genre
    fig = px.bar(df[mask],x="Genre", y="EU_Sales")
    # print(value_Genre)
    return fig

if __name__=='__main__':
    app.run_server()
```

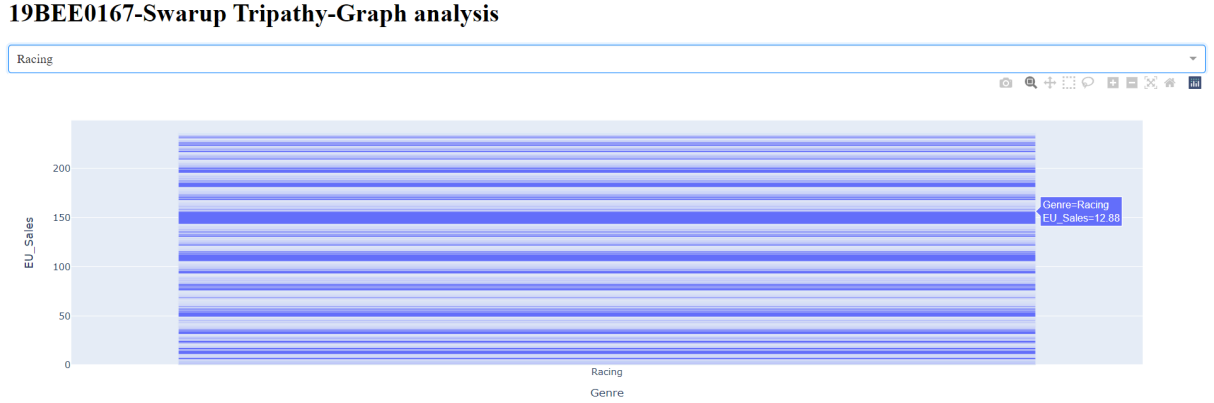
7.1 Code output

```
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
```









Similarly, we get the plots for other genres as well being updated when we select our specific genre from the dropdown.

For Covid 19 data taken from Kaggle Dataset csv

8 Python Code

```
import dash
import pandas as pd
import plotly.express as px

from dash import html
from dash import dcc
from dash.dependencies import Input, Output

df=pd.read_csv("C:/Users/iwill/Desktop/VIT/Semester 6/IoT Lab/Lab 5/covid_19_india.csv")
print(df)
print("################################################################################")
print(df[:5])  # to display first 5 data
```

8.1 Code output

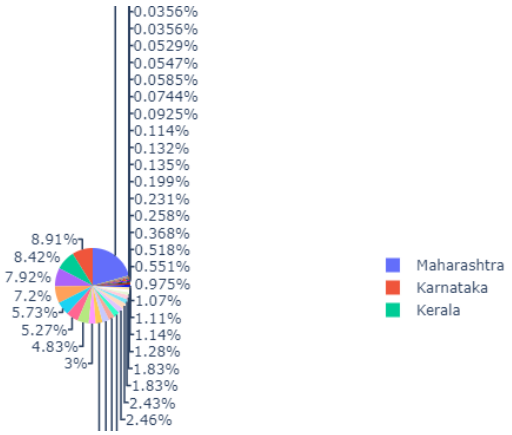
```
         Sno        Date     Time          State ConfirmedIndianNational  \
0          1  30-01-2020  6:00 PM         Kerala                       1
1          2  31-01-2020  6:00 PM         Kerala                       1
2          3  01-02-2020  6:00 PM         Kerala                       2
3          4  02-02-2020  6:00 PM         Kerala                       3
4          5  03-02-2020  6:00 PM         Kerala                       3
...      ...         ...      ...            ...                     ...
18105  18106  11-08-2021  8:00 AM      Telangana                       -
18106  18107  11-08-2021  8:00 AM        Tripura                       -
18107  18108  11-08-2021  8:00 AM    Uttarakhand                       -
18108  18109  11-08-2021  8:00 AM  Uttar Pradesh                       -
18109  18110  11-08-2021  8:00 AM    West Bengal                       -

      ConfirmedForeignNational    Cured  Deaths  Confirmed
0                            0        0       0          1
1                            0        0       0          1
2                            0        0       0          2
3                            0        0       0          3
4                            0        0       0          3
...                        ...      ...     ...        ...
18105                        -   638410    3831     650353
18106                        -    77811     773      80660
18107                        -   334650    7368     342462
18108                        -  1685492   22775    1708812
18109                        -  1506532   18252    1534999

[18110 rows x 9 columns]
################################################################################
   Sno        Date     Time   State ConfirmedIndianNational  \
0    1  30-01-2020  6:00 PM  Kerala                       1
1    2  31-01-2020  6:00 PM  Kerala                       1
2    3  01-02-2020  6:00 PM  Kerala                       2
```

```
3    4  02-02-2020  6:00 PM  Kerala                        3
4    5  03-02-2020  6:00 PM  Kerala                        3

   ConfirmedForeignNational  Cured  Deaths  Confirmed
0                         0      0       0          1
1                         0      0       0          1
2                         0      0       0          2
3                         0      0       0          3
4                         0      0       0          3
```

9 Python Code

```
fig_pie=px.pie(data_frame=df, names="State", values="Confirmed")
fig_pie.show()
```

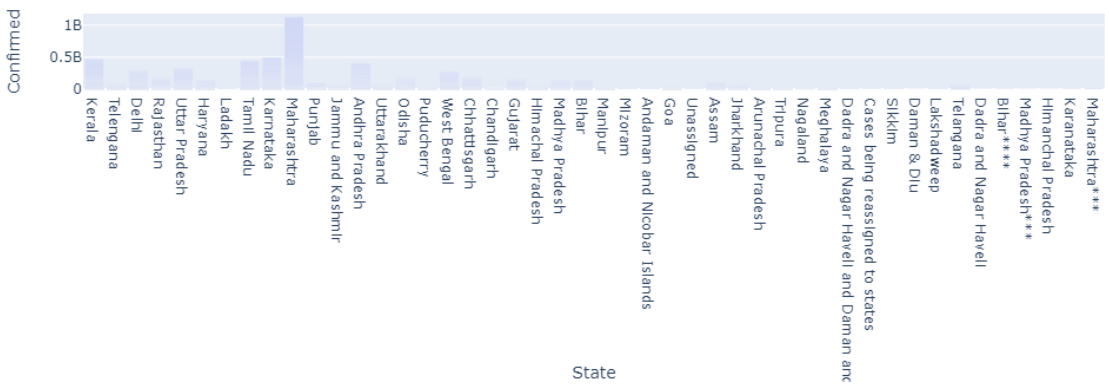
9.1 Code output



10 Python Code

```
fig_bar = px.bar(data_frame=df,x="State", y="Confirmed")
fig_bar.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig_bar.show()
```

10.1 Code output



11 Final Python Code on DASH application

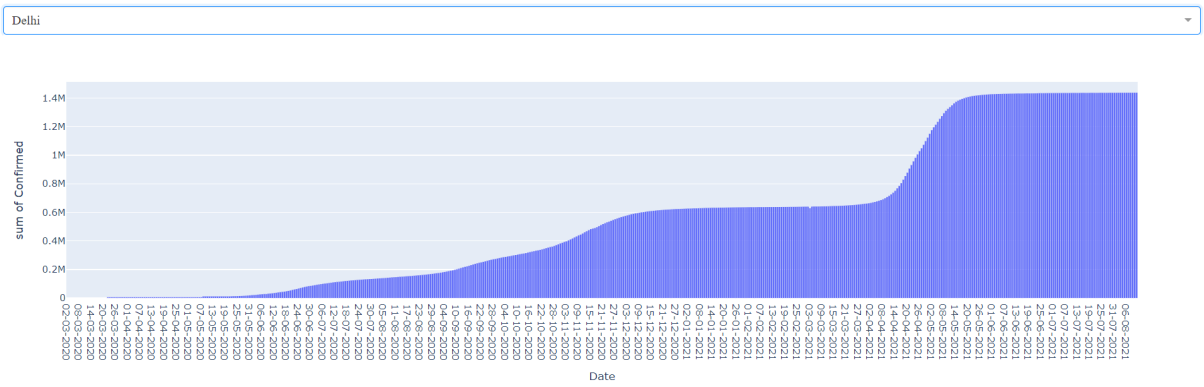
```
import dash
import plotly.express as px
import pandas as pd
from dash import html
from dash import dcc
from dash.dependencies import Input, Output

# import dash_core_components as dcc
# import dash_html_components as html

df = pd.read_csv("C:/Users/iwill/Desktop/VIT/Semester 6/IoT Lab/Lab 5/covid_19_india.csv")


gen = df.State.unique()

app = dash.Dash(__name__)
app.layout=html.Div([
    html.H1("19BEE0167-Swarup Tripathy-Graph analysis for COVID-19"),
    dcc.Dropdown(
        id='State-choice',
        options =[{'label':x,'value':x} for x in gen],
        value=gen[0],
        clearable=False),
    dcc.Graph(id="bar-chart")
])

@app.callback(
    Output(component_id="bar-chart", component_property='figure'),
    Input(component_id='State-choice', component_property='value')
)

def interactive_graphing(State):
    mask = df["State"] == State
    fig = px.histogram(df[mask],x="Date", y="Confirmed")
    # print(value_Genre)
    return fig

if __name__=='__main__':
    app.run_server()
```

11.1 Code output

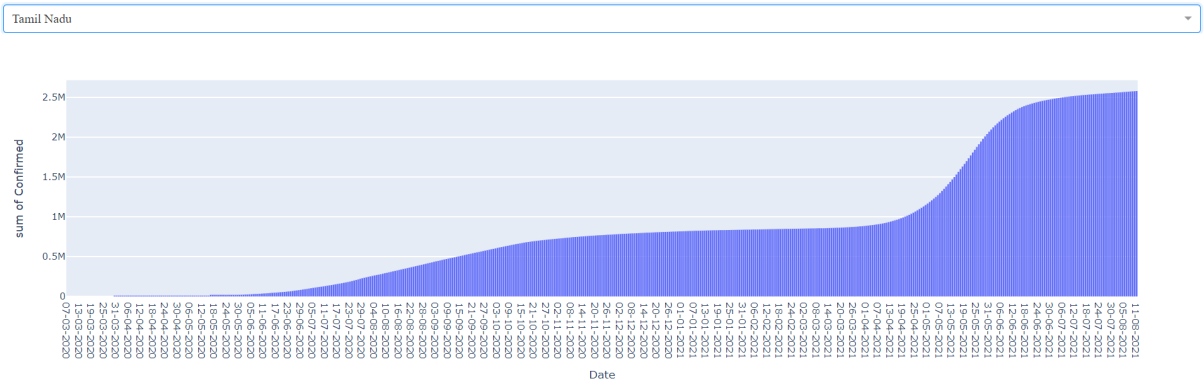
```
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
```

```
  Use a production WSGI server instead.
* Debug mode: off
```

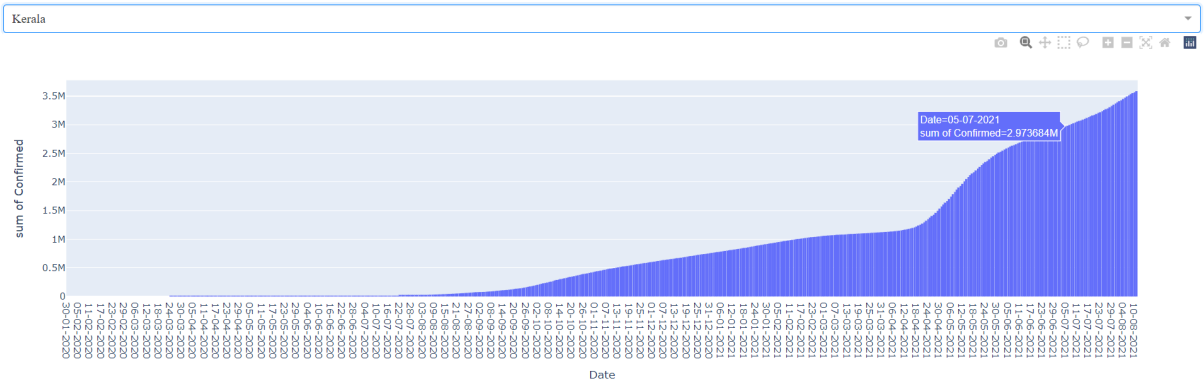
19BEE0167-Swarup Tripathy-Graph analysis for COVID-19

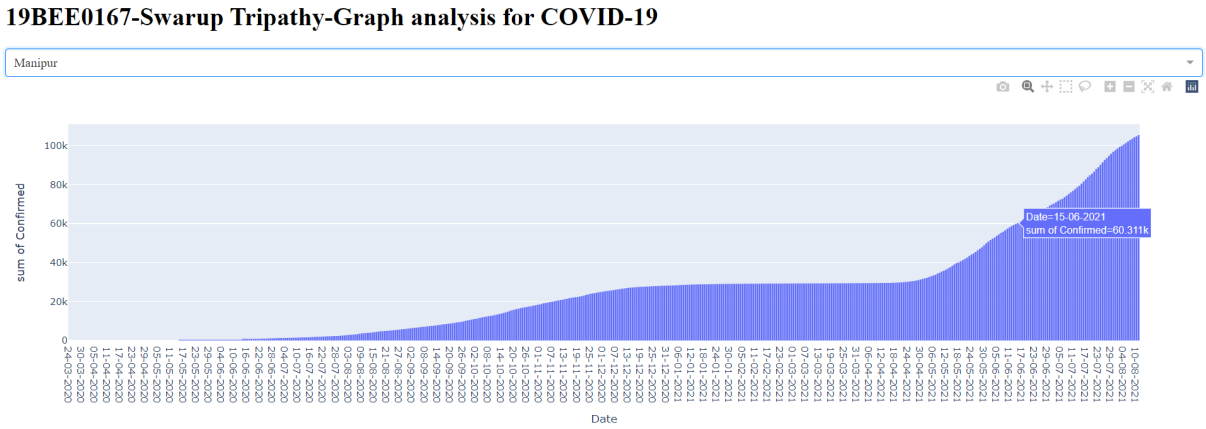


19BEE0167-Swarup Tripathy-Graph analysis for COVID-19



19BEE0167-Swarup Tripathy-Graph analysis for COVID-19





Similarly, we get the plots for other genres as well being updated when we select our specific genre from the dropdown.

Name: **Swarup Tripathy**
Course: **ECE3502: IoT Domain Analyst**

Assignment Number: **3**
Date: March 6, 2022

Reg No: 19BEE0167

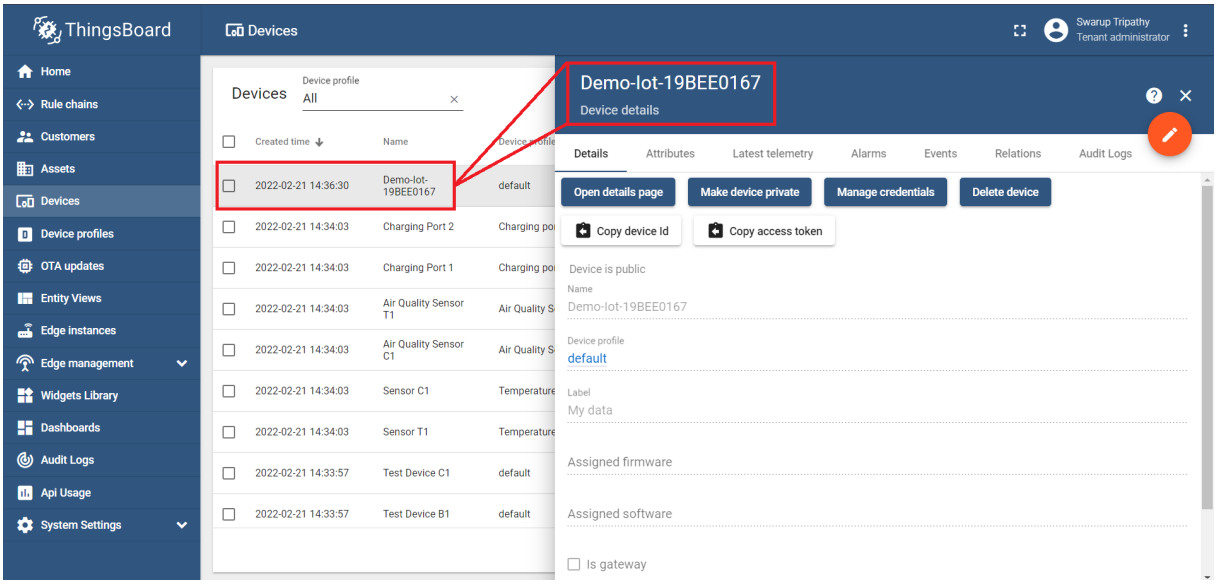
1 Problem 2

Online Dashboard creation and to push the data into Thingsboard using http protocol.

B. To push the data into Thingsboard using mqtt protocol and Create Dashboard in Thingsboard using MQTT. Also create two devices and feed data through python and compare the values of the devices using various widgets in thingsboard

Configuration of Thingsboard

1. Creating an account on Thingsboard
2. First we need to create a device
   - Name → Demo-Iot-19BEE0167
   - Device type → default
   - Label → my data
3. When clicked on 'Device', and when we select 'latest telemetry' it provides the last data recorded
4. We make sure that we copy the 'token'
5. Creating a Dashboard
   - In the dashboards section we create a new dashboard for our device named 'Demo-Iot-19BEE0167'
   - Click on 'Open Dashboard'
   - Now we add a new widget where our data will be displayed
   - For this experiment, i have added a digital guage and a graph to show the data simultaneously on both.
6. Now it is time to look at the python code for pushing the data to thingsboard for visualisation.



2 Python Code

1. Installing paho on google colab with the following code

```
pip install paho-mqtt
```

2. Importing the necessary libraries

```
##########################################################################################
import os
import time
import sys
import paho.mqtt.client as mqtt
import json
import random

Thingsboard_host = 'demo.thingsboard.io'
##########################################################################################
```

3. Now we need to push random data in the range between 0 to 180 for which we are inserting it in between a while loop to send data after every 5 seconds of delay.

```
##########################################################################################
access_token = 'JVW6vmIzxX0YSJ0eqYzv'
sensor_data = {'Mysensor':0}

# for i in range(10):
#   sensor_data={'Mysensor':random.randint()}

# client = mqtt.Client('python')
client = mqtt.Client()
client.username_pw_set(access_token)
```

```
client.connect(Thingsboard_host, 1883, 60)
client.loop_start() # starting a connection to the devices

try:
  while True:
    senval = random.randrange(0,180)  # randomly selected element from the
                                        specified range.
    print(senval)
    sensor_data['Mysensor']=senval    # since 'Mysensor' is the key so we access
                                        the key to feed the value
    client.publish('v1/devices/me/telemetry',json.dumps(sensor_data),1)
    time.sleep(3)   # giving a time delay of 5s

except KeyboardInterrupt:
  pass

client.loop_stop()
client.disconnect()
#############################################################################
```

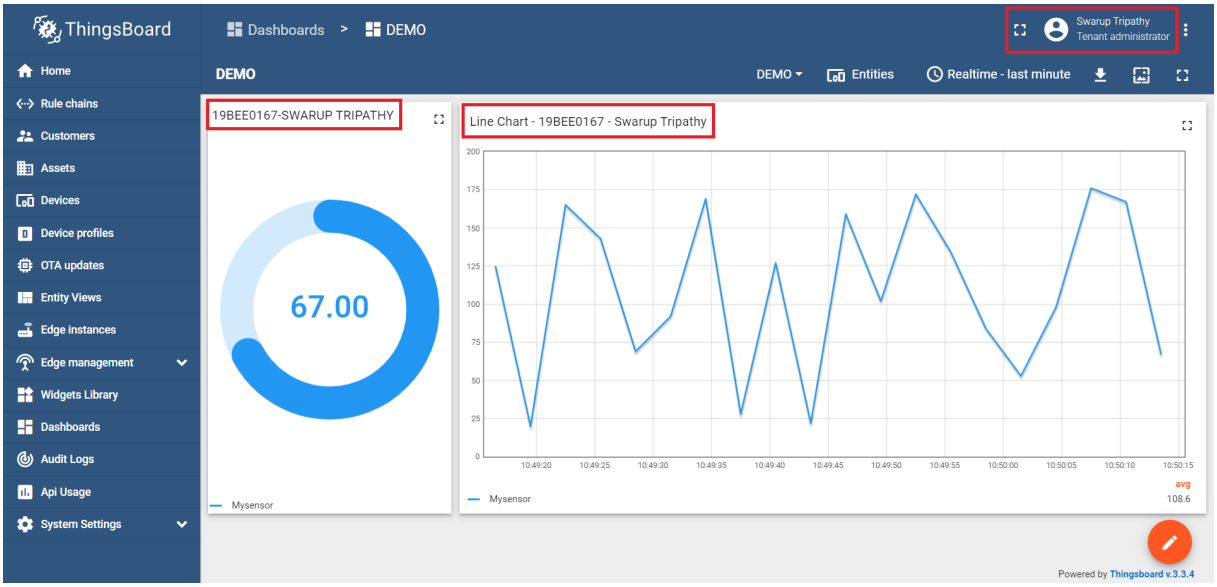
4. In Python, pass is a null statement. The interpreter does not ignore a pass statement, but nothing happens and the statement results into no operation.The pass statement is useful when you don't write the implementation of a function but you want to implement it in the future.

2.1 Code output

```
51
83
71
125
20
165
143
69
92
169
28
127
22
159
102
172
134
84
.
.
.
32
152
49
151
```

```
76
144
73
40
```

1. Below you can see the output graph that is updating realtime when we run the code
2. Following, 2 widgets have been displayed i.e., a digital guage and a line chart and the code output can be compared with the data on line chart for relevancy



3. Below I have also displated the 'latest telemetry'