```
from google.colab import drive
drive.mount('/content/drive')
```

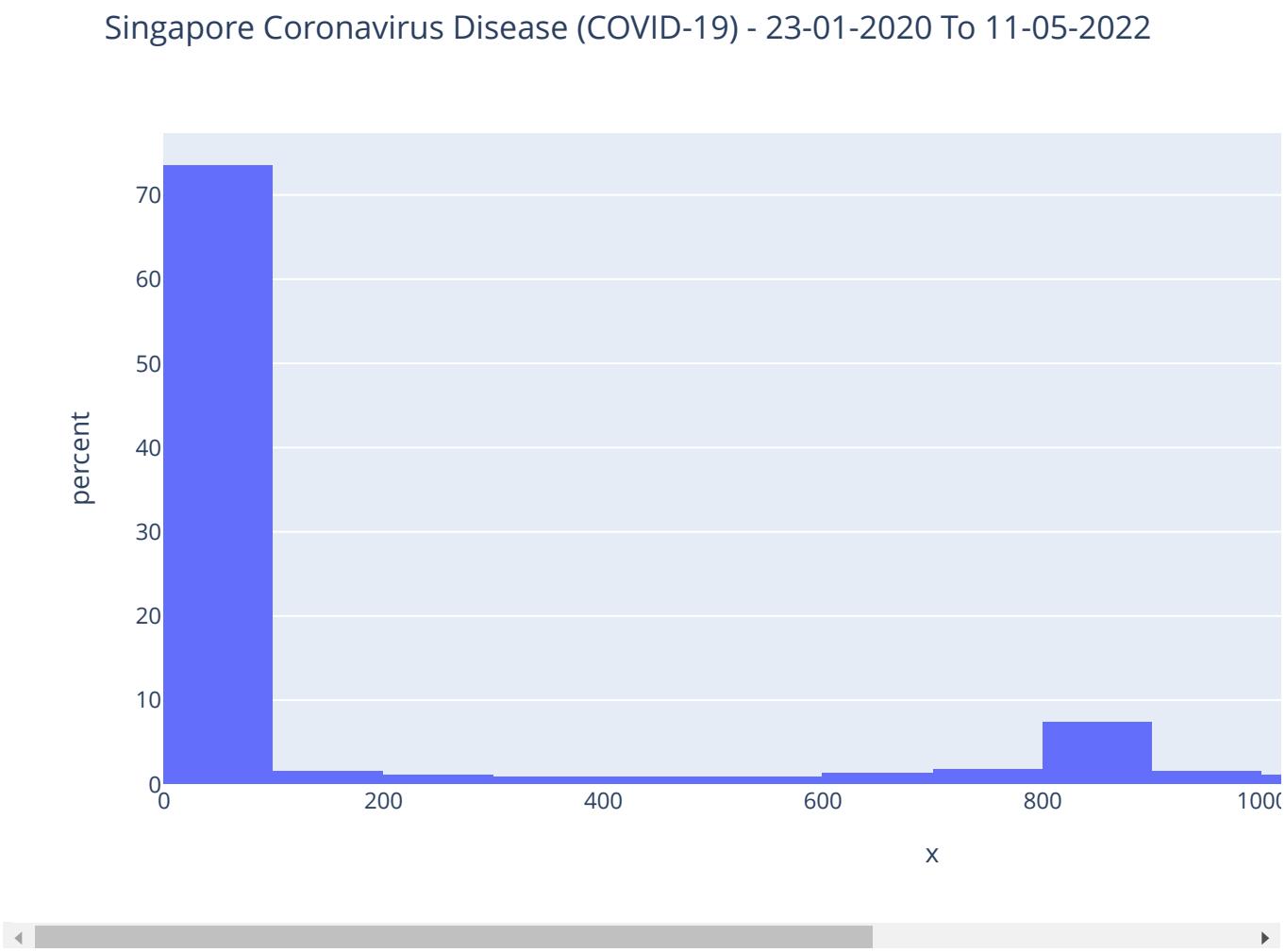
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
from pandas.core import frame
import os
import pandas as pd
import json
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# To check what is the directory listing, we use the built-in function .listdir
os.listdir('/content/drive')

folder_path = '/content/drive/My Drive/Gov_Tech_Task/Data'
#print(os.listdir('/content/drive/My Drive/Gov_Tech_Task/Data'))
file_path = folder_path + '/Q4-Data_Singapore_full.json'


openfile=open(file_path)
jsondata=json.load(openfile)
df=pd.DataFrame(jsondata)

openfile.close()
#print(df)

main_df_covid = df[["Country","CountryCode","Confirmed","Deaths","Recovered","A
main_df_covid.head()
```

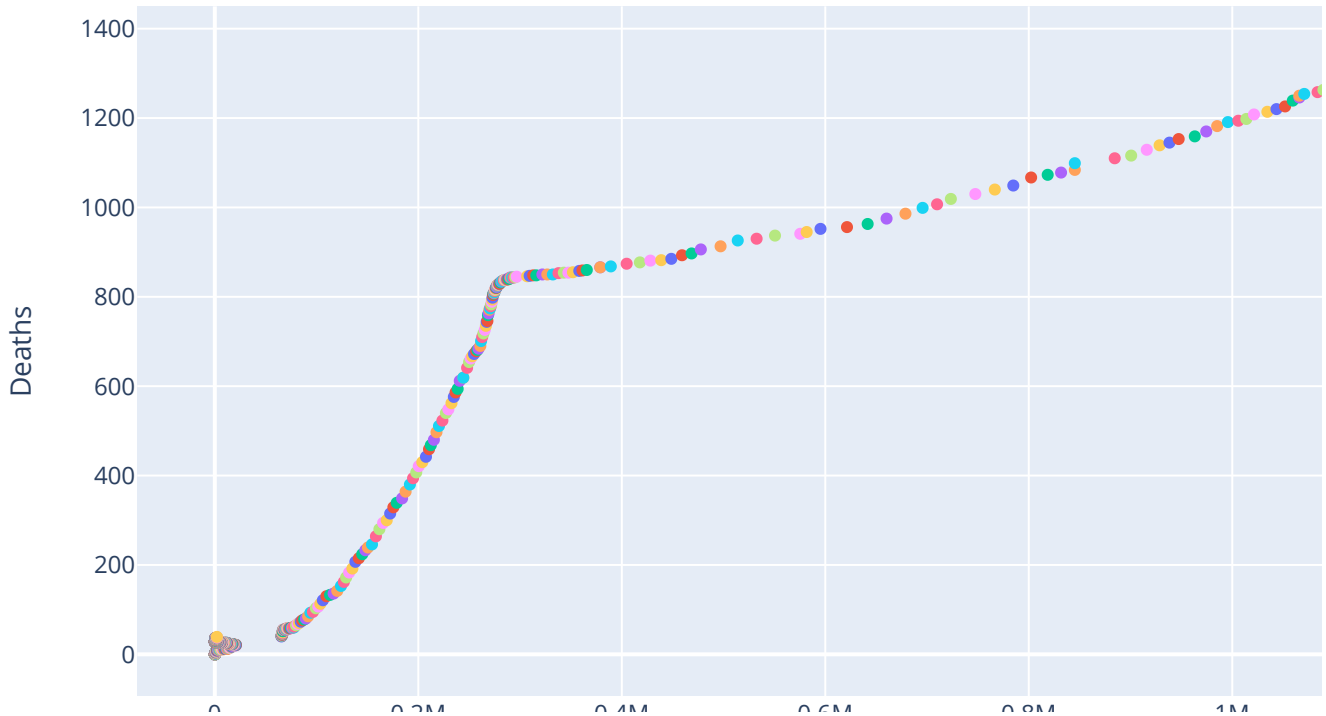
	Country	CountryCode	Confirmed	Deaths	Recovered	Active	Date
0	Singapore	SG	1	0	0	1	2020-01-23T00:00:00Z
1	Singapore	SG	3	0	0	3	2020-01-24T00:00:00Z
2	Singapore	SG	3	0	0	3	2020-01-25T00:00:00Z
3	Singapore	SG	4	0	0	4	2020-01-26T00:00:00Z
4	Singapore	SG	5	0	0	5	2020-01-27T00:00:00Z

```
import plotly.express as px
df = px.histogram(x=main_df_covid['Deaths'],
                  histnorm='percent',
```

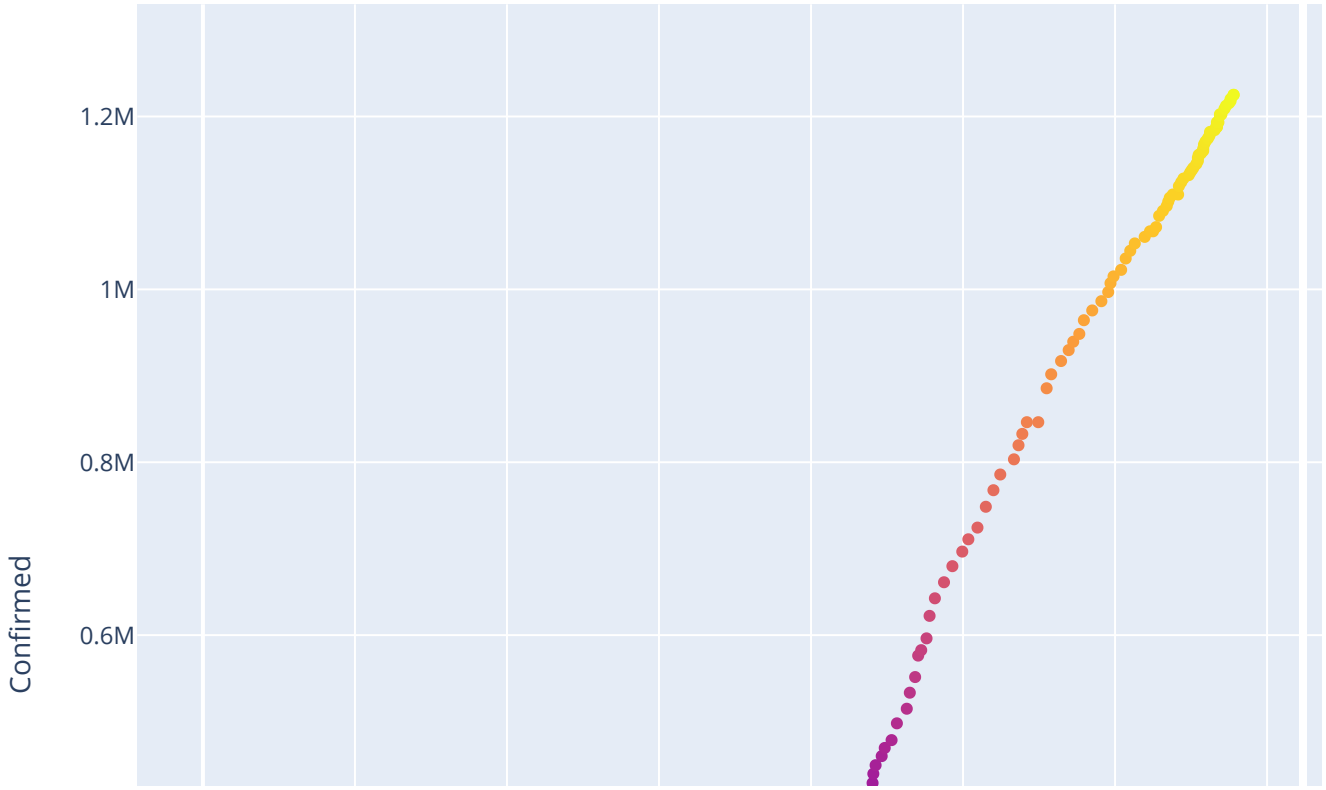
```
                    nbins=25,
                    title='Singapore Coronavirus Disease (COVID-19) - 23-01-
                    )
df.show()
```




```
import plotly.express as px

fig = px.scatter(main_df_covid,
                 x='Active',
                 y='Deaths',
                 color='Date',
                 hover_name= "Confirmed" ,
                 title='Singapore Coronavirus Disease (COVID-19) - 23-01-2020 To
                )
fig.update_traces(mode="markers+lines")
fig.show()
```




```
1 import plotly.express as px
2 df = main_df_covid
3
4 fig = px.scatter(df, x="Deaths", y="Confirmed", color="Active",
5                  marginal_y="violin",height=800,
6                  title='Singapore Coronavirus Disease (COVID-19) - 23-01-2021 T
7 )
8 fig.show()
```




```
def ecdf(data):
    """Compute ECDF and generate a dataframe."""
    df = pd.DataFrame()
    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    # sort the data
    df['x'] = np.sort(data)

    # y-data for the ECDF: y
    # y goes from 1/n to 1 in equal spaced intervals, arange() does that
    # but arange end value is not inclusive, so n+1
    df['y'] = np.arange(1 , n+1) / n

    return df
```

```
import numpy as np
df_ecdf = ecdf(main_df_covid['Deaths'])
df_ecdf
# Get Mean DF
```

```
df_mean = main_df_covid['Deaths'].mean()
# Get STD DF
df_std = main_df_covid['Deaths'].std()
# Get Total Cases DF
df_cases = np.random.normal(df_mean, df_std, size=10000)

# apply ecdf function on sampled (theoretical normal) data
df_total_cases = ecdf(df_cases)

# plot the scatter chart
df_fig = px.scatter(df_ecdf, x='x', y='y')
df_fig.add_scatter(x=df_total_cases.x, y=df_total_cases.y, name='Theoretical E(
df_fig.show()
```