

Task

(a) Load the data that was pre-processed from the implemented data warehouse (PostgreSQL) in assignment three (3 Marks)

I made sure I have several libraries installed that might have not been installed before

- pip install psycopg2 - was installed before as it allows connection with the PostgreSQL
- ipython-sql - this enables the use of SQL magic functions that contain % and %% , allowing you to write SQL style code right in Jupyter Notebook.
- sqlalchemy - it will mainly be used to store SQL queries into a pandas dataframe.

```
In [191]: 1 #to load ipython-sql
          2 %load_ext sql
```

```
In [237]: 1 from sqlalchemy import create_engine
          2 import pandas as pd
          3 import matplotlib.pyplot as plt
          4 import seaborn as sns
          5 from sklearn.preprocessing import MinMaxScaler
          6 from sklearn.cluster import KMeans
```

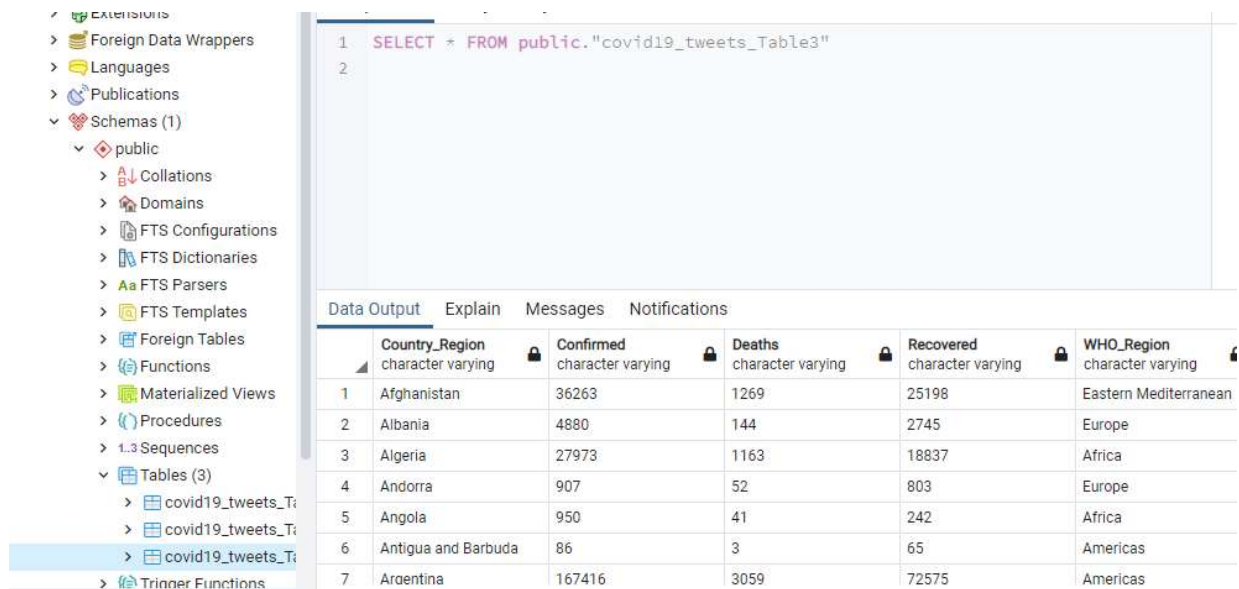
Connect ipython-sql to my database

```
In [238]: 1 %sql postgresql://postgres:1235@localhost/covid19db
```

To connect sqlalchemy to the database

```
In [239]: 1 engine = create_engine('postgresql://postgres:1235@localhost/covid19db')
```

Below is a sample of the database we want to query



	Country_Region character varying	Confirmed character varying	Deaths character varying	Recovered character varying	WHO_Region character varying
1	Afghanistan	36263	1269	25198	Eastern Mediterranean
2	Albania	4880	144	2745	Europe
3	Algeria	27973	1163	18837	Africa
4	Andorra	907	52	803	Europe
5	Angola	950	41	242	Africa
6	Antigua and Barbuda	86	3	65	Americas
7	Argentina	167416	3059	72575	Americas

Database querying. The SQL code should be in its own block, separate from Python code

In [240]:

```
1 %%sql
2
3 SELECT
4     *
5 FROM
6     public."covid19_tweets_Table3" LIMIT 7
```

* postgresql://postgres:***@localhost/covid19db
7 rows affected.

Out[240]:

	Country_Region	Confirmed	Deaths	Recovered	WHO_Region
	Afghanistan	36263	1269	25198	Eastern Mediterranean
	Albania	4880	144	2745	Europe
	Algeria	27973	1163	18837	Africa
	Andorra	907	52	803	Europe
	Angola	950	41	242	Africa
	Antigua and Barbuda	86	3	65	Americas
	Argentina	167416	3059	72575	Americas

Store the query in a pandas DataFrame

In [241]:

```
1 df = pd.read_sql('SELECT * FROM public."covid19_tweets_Table3"', engine)
2 df.sample(7)
```

Out[241]:

	Country_Region	Confirmed	Deaths	Recovered	WHO_Region
102	Luxembourg	6321	112	4825	Europe
10	Azerbaijan	30446	423	23242	Europe
154	South Africa	452529	7067	274925	Africa
47	Denmark	13761	613	12605	Europe
170	Trinidad and Tobago	148	8	128	Americas
122	Nicaragua	3439	108	2492	Americas
99	Libya	2827	64	577	Eastern Mediterranean

Check if there are any null values

In [242]:

```
1 df.isnull().sum(axis = 0)
```

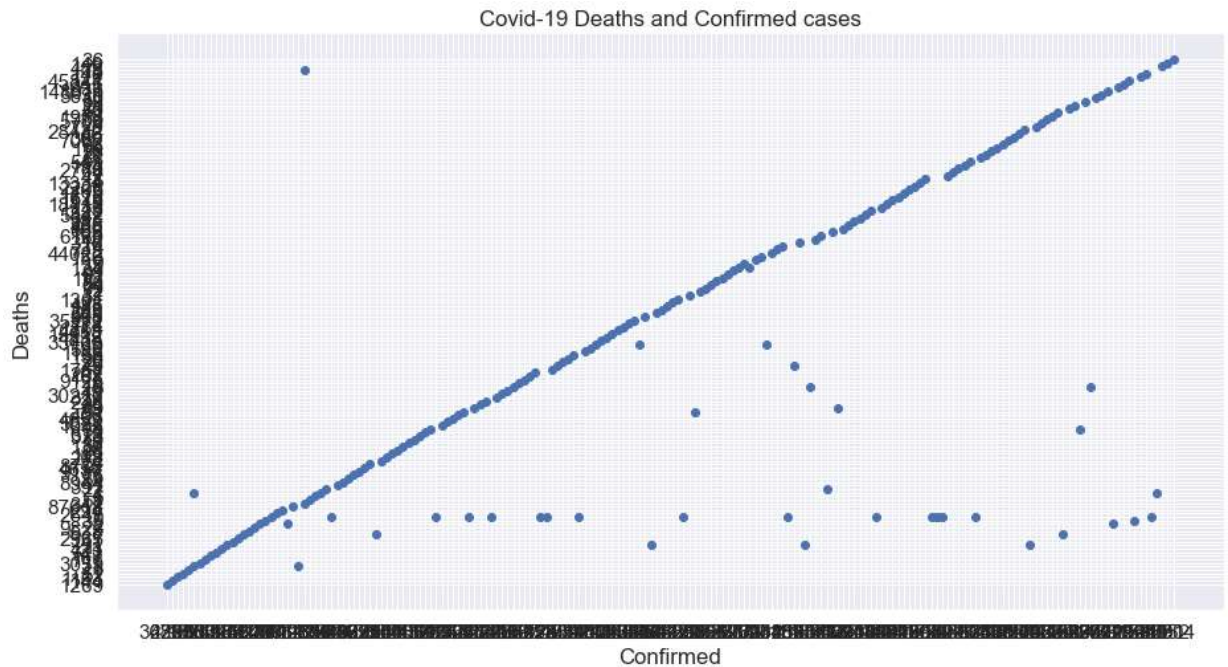
Out[242]:

```
Country_Region    0
Confirmed         0
Deaths           0
Recovered         0
WHO_Region        0
dtype: int64
```

Plot a scatter plot to have a glimpse of the data

```
In [243]: 1 plt.figure(figsize=(15, 8))
2 plt.scatter(df['Confirmed'], df['Deaths'])
3 plt.title("Covid-19 Deaths and Confirmed cases") #title
4 plt.xlabel("Confirmed") #x Label
5 plt.ylabel("Deaths") #y Label
```

Out[243]: Text(0, 0.5, 'Deaths')



Scale the data to make it look more presentable

The MinMaxScaler will make sure the values are scaled to between 0 and 1

```
In [244]: 1 scaler = MinMaxScaler()
2 scaler.fit(df[['Confirmed', 'Deaths']])
3 df[['Confirmed', 'Deaths']] = scaler.transform(df[['Confirmed', 'Deaths']])
4 df
```

```
Out[244]:
```

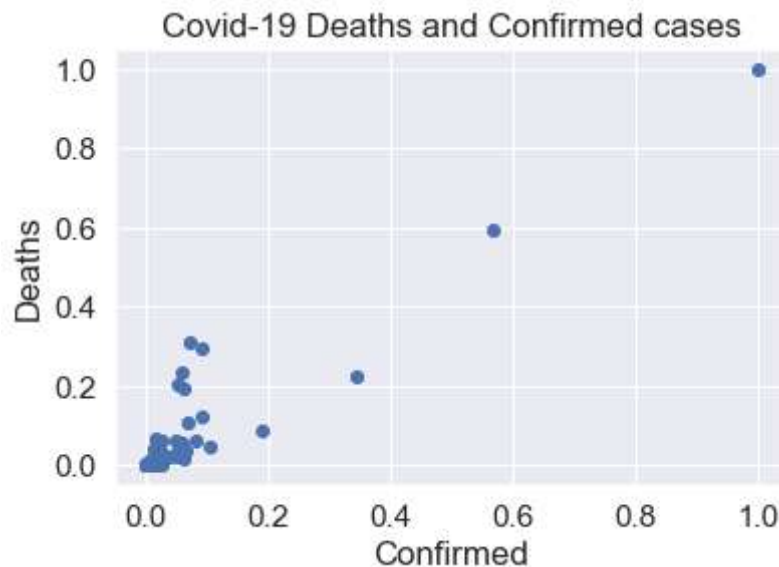
	Country_Region	Confirmed	Deaths	Recovered	WHO_Region
0	Afghanistan	0.00845	0.008574	25198	Eastern Mediterranean
1	Albania	0.001135	0.000973	2745	Europe
2	Algeria	0.006518	0.007858	18837	Africa
3	Andorra	0.000209	0.000351	803	Europe
4	Angola	0.000219	0.000277	242	Africa
...
182	West Bank and Gaza	0.002473	0.000527	3752	Eastern Mediterranean
183	Western Sahara	0.0	0.000007	8	Africa
184	Yemen	0.000392	0.003263	833	Eastern Mediterranean
185	Zambia	0.001059	0.000946	2815	Africa
186	Zimbabwe	0.000628	0.000243	542	Africa

187 rows × 5 columns

Visualize the results of the scaled data

```
In [245]: 1 plt.scatter(df['Confirmed'], df['Deaths'])
2 plt.title("Covid-19 Deaths and Confirmed cases") #title
3 plt.xlabel("Confirmed") #x Label
4 plt.ylabel("Deaths") #y Label
```

```
Out[245]: Text(0, 0.5, 'Deaths')
```



(b) Use python libraries to perform KMeans clustering on the data (5 Marks)


```
In [251]: 1 km.cluster_centers_
```

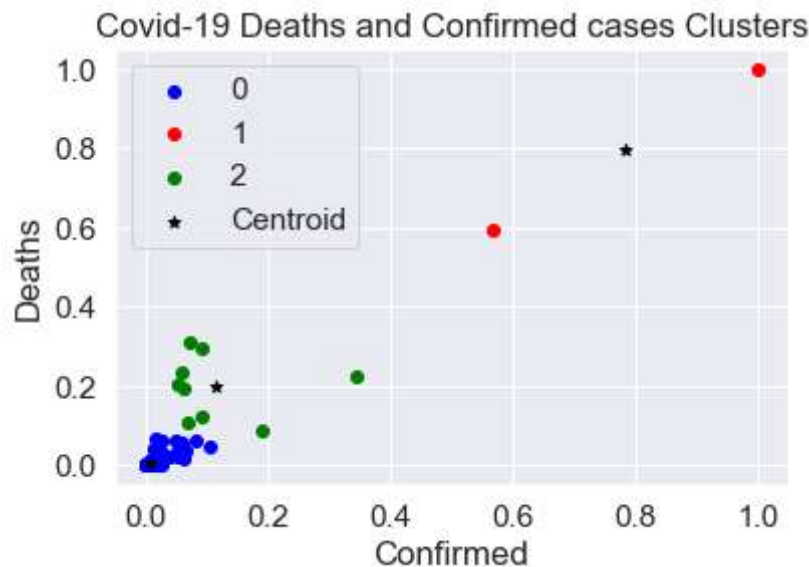
```
Out[251]: array([[0.0070585 , 0.00590071],
                [0.7846414 , 0.79598476],
                [0.11437414, 0.19870445]])
```

(c) Use visualization techniques to represent the discovered clusters (4 Marks)

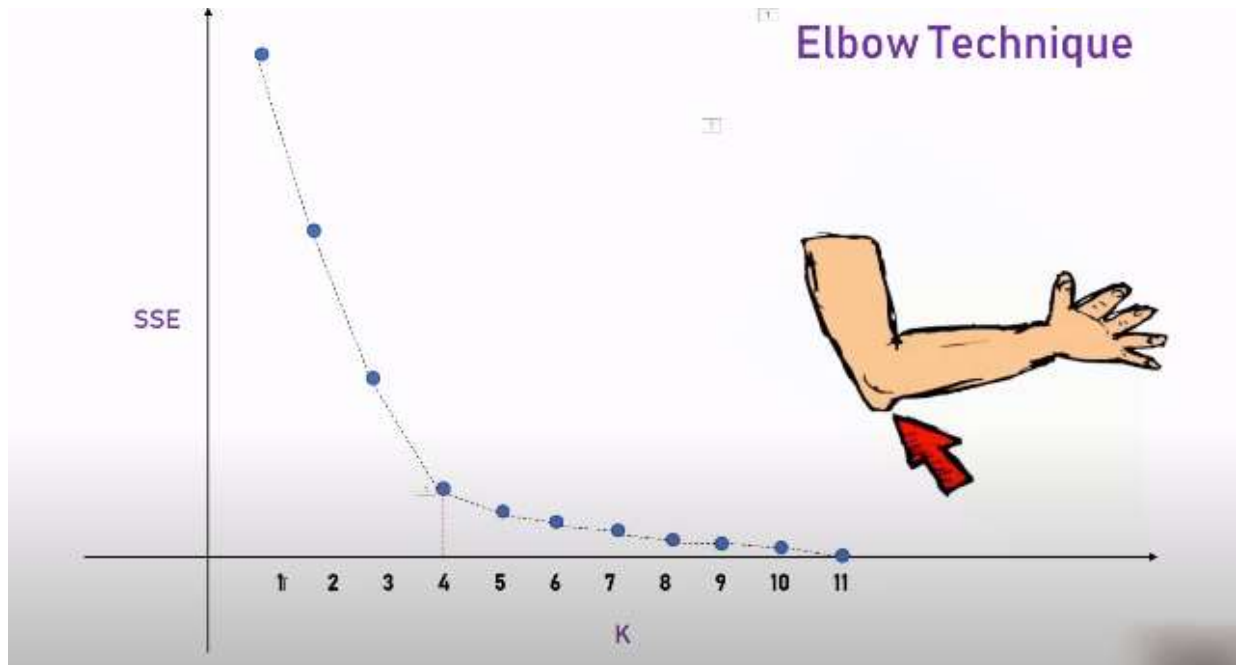
Plot the clusters on a scatter plot

```
In [252]: 1 df1=df[df.Cluster==0]
          2 df2=df[df.Cluster==1]
          3 df3=df[df.Cluster==2]
          4
          5 plt.scatter(df1.Confirmed,df1.Deaths,color='blue',label='0')
          6 plt.scatter(df2.Confirmed,df2.Deaths,color='red',label='1')
          7 plt.scatter(df3.Confirmed,df3.Deaths,color='green',label='2')
          8
          9 plt.scatter(km.cluster_centers_[0],km.cluster_centers_[1],color='black',
         10
         11 plt.title("Covid-19 Deaths and Confirmed cases Clusters") #title
         12 plt.xlabel("Confirmed") #x Label
         13 plt.ylabel("Deaths") #y Label
         14 plt.legend()
```

```
Out[252]: <matplotlib.legend.Legend at 0xef13433bb0>
```



For huge amounts of data, we use the elbow method to find the optimal value of k



```
In [253]: 1 k_rng = range(1,10)
          2 sse = []
          3 for k in k_rng:
          4     km = KMeans(n_clusters=k)
          5     km.fit(df[['Confirmed', 'Deaths']])
          6     #inertia gives the sum of squared error
          7     sse.append(km.inertia_)
```

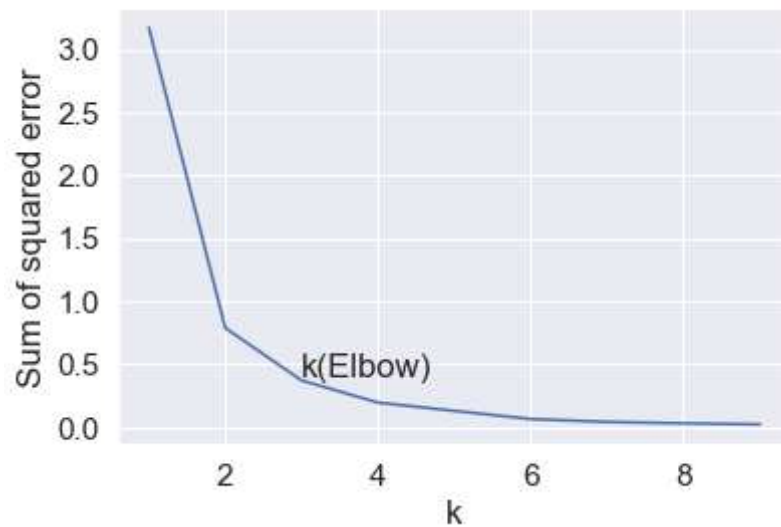
C:\Users\user\anaconda3\lib\site-packages\sklearn\cluster_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(

```
In [254]: 1 sse
```

```
Out[254]: [3.172764332587553,
           0.7864767213169597,
           0.36958563147365586,
           0.1935825439317292,
           0.12784756906070446,
           0.0630115430287818,
           0.03999993898814132,
           0.029065897745118512,
           0.020413243382037705]
```

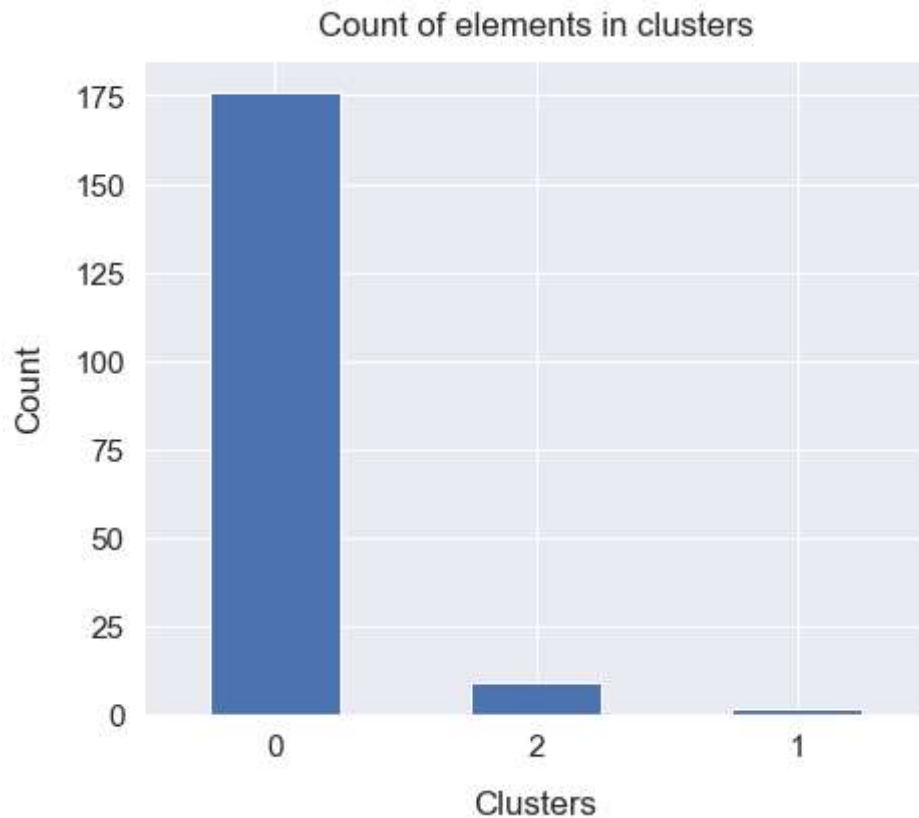
SSE was very high to begin with then it went reducing
Let's now plot to find K(the elbow and check whether it is same as what we had)

```
In [255]: 1 plt.xlabel('k')
          2 plt.ylabel('Sum of squared error')
          3 plt.plot(k_rng,sse)
          4 plt.annotate("k(Elbow)",(3,0.4))
          5 plt.show()
```



Visualize the count of clusters on a bar graph


```
In [256]: 1 sns.set(font_scale=1.4)
2 df['Cluster'].value_counts().plot(kind='bar', figsize=(7, 6), rot=0)
3 plt.xlabel("Clusters", labelpad=14)
4 plt.ylabel("Count", labelpad=14)
5 plt.title("Count of elements in clusters", y=1.02);
```

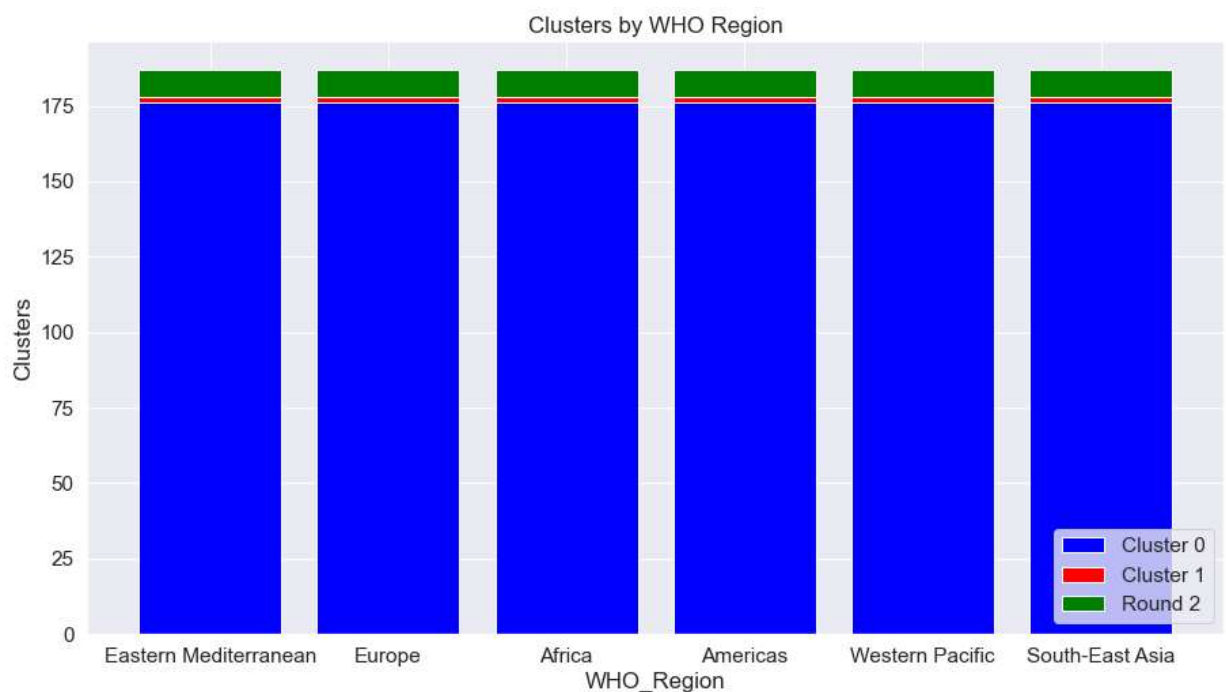


Visualize the clusters by WHO_Region on a stacked barchat

```

In [257]: 1 x = df.WHO_Region
2 y1 = (df.Cluster==0).sum()
3 y2 = (df.Cluster==1).sum()
4 y3 = (df.Cluster==2).sum()
5
6 # plot bars in stack manner
7 plt.figure(figsize=(15, 8))
8 plt.bar(x, y1, color='blue',)
9 plt.bar(x, y2, bottom=y1, color='red')
10 plt.bar(x, y3, bottom=y1+y2, color='green')
11
12 plt.xlabel("WHO_Region")
13 plt.ylabel("Clusters")
14 plt.legend(["Cluster 0", "Cluster 1", "Round 2"])
15 plt.title("Clusters by WHO Region")
16 plt.show()

```



(d) Describe characteristics and suggested names for the developed clusters. (3 Marks)

CLUSTER 0

- Most Countries in the WHO Region belong to this cluster
- Very many countries have low cases of Deaths and Recoveries
- Can be called a low risk COVID-19 Region

CLUSTER 1

- Third in terms of number countries belonging to this cluster
- Very few countries but high cases of Deaths and Confirmations
- Can be called a high risk COVID-19 Region

CLUSTER 2

- Second in terms of number countries belonging to this cluster
- Can be called a median risk COVID-19 Region

(e) Discuss potential applications of the extracted clusters (5 Marks)

This data can be applied by governments, health practitioners, and researchers to help make decisions that help in prevention and control of covid 19

Some of the measures according to region are discussed below

CLUSTER 0- Low Risk

- Make sure people are fully vaccinated before travel to these destinations.
- Educate people on control measures and ensure strict adherence to those measures

CLUSTER 1 - High Risk

- Make sure people are fully vaccinated before traveling to these destinations.
- Unvaccinated travelers should avoid nonessential travel to these destinations.
- Introduce strict measures such as lockdown in these areas to reduce the spread.
- Heavily invest on resources that will aid in curbing the effects of the disease as well as prevent more occurrences.

CLUSTER 2 - Median Risk

- Make sure people are fully vaccinated before traveling to these destinations.
- Unvaccinated travelers who are at increased risk for severe illness from COVID-19 should avoid nonessential travel to the these destinations.
- Implement measures like partial lockdown

References

PostgreSQL Integration with Jupyter Notebook - <https://medium.com/analytics-vidhya/postgresql-integration-with-jupyter-notebook-deb97579a38d> (<https://medium.com/analytics-vidhya/postgresql-integration-with-jupyter-notebook-deb97579a38d>).

Machine Learning Tutorial Python - 13: K Means Clustering Algorithm - <https://www.youtube.com/watch?v=EltlUEPClzM> (<https://www.youtube.com/watch?v=EltlUEPClzM>).

Bar Plot using Pandas - <https://dfrieds.com/data-visualizations/bar-plot-python-pandas.html> (<https://dfrieds.com/data-visualizations/bar-plot-python-pandas.html>).

How to Deploy a Machine Learning Model (K Means) and Generate Insights - https://www.youtube.com/watch?v=XjKNxgc4Is4&list=RDCMUCqtValEjW3-Mar0sqiWO_Hw&index=2 (https://www.youtube.com/watch?v=XjKNxgc4Is4&list=RDCMUCqtValEjW3-Mar0sqiWO_Hw&index=2).

Create a stacked bar plot in Matplotlib - [geeksforgeeks.org/create-a-stacked-bar-plot-in-matplotlib/](https://www.geeksforgeeks.org/create-a-stacked-bar-plot-in-matplotlib/)

COVID-19 Travel Recommendations by Destination - <https://www.cdc.gov/coronavirus/2019-ncov/travelers/map-and-travel-notice.html> (<https://www.cdc.gov/coronavirus/2019-ncov/travelers/map-and-travel-notice.html>).