library session pandas covid

April 14, 2022

Identitas:

[2]:
```
print('NAMA:')
print('Muhammad Ogin Hasanuddin')
```

```
NAMA:
Ghiffary Rifqialdi
```

Lakukan instruksi di bawah ini dengan menggunakan dataset yang telah disediakan!

[1]:
```
#Importing required modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Reading datasets
#main dataset
df = pd.read_csv('covid.csv', index_col=0)
#region and climate dataset
df_cat = pd.read_csv('covid_cat.csv')

# dapatkan data dari https://github.com/muhammadoginh/data_covid
# letakkan covid.csv dan covid_cat.csv dalam satu folder bersama project ipynb
```

[4]:
```
df.head()
```

[4]:

	country	province	date	confirmed	recovered	deaths
0	Afghanistan	0	2020-01-22	0.0	0.0	0.0
1	Albania	0	2020-01-22	0.0	0.0	0.0
2	Algeria	0	2020-01-22	0.0	0.0	0.0
3	Andorra	0	2020-01-22	0.0	0.0	0.0
4	Angola	0	2020-01-22	0.0	0.0	0.0

[5]:
```
df.tail()
```

[5]:

	country	province	date	confirmed	recovered	deaths
175835	Canada	0	2021-10-06	NaN	0.0	NaN
175836	Canada	0	2021-10-07	NaN	0.0	NaN
175837	Canada	0	2021-10-08	NaN	0.0	NaN
175838	Canada	0	2021-10-09	NaN	0.0	NaN

```
175839  Canada        0  2021-10-10        NaN        0.0      NaN
```

[59]:
```
df.date = pd.to_datetime(df.date, format='%Y-%m-%d')
```

[54]:
```
df.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 175840 entries, 0 to 175839
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   country    175840 non-null  object
 1   province   175840 non-null  object
 2   date       175840 non-null  datetime64[ns]
 3   confirmed  175212 non-null  float64
 4   recovered  165792 non-null  float64
 5   deaths     175212 non-null  float64
dtypes: datetime64[ns](1), float64(3), object(2)
memory usage: 9.4+ MB
```

[60]:
```
df_cat.head()
```

[60]:

	country	region	climate
0	Afghanistan	Asia & Pacific	nontropic
1	Albania	Europe	nontropic
2	Algeria	Arab States	nontropic
3	Andorra	Europe	nontropic
4	Angola	Africa	tropic

[8]:
```
df_cat.tail()
```

[8]:

	country	region	climate
194	Vatican	Europe	nontropic
195	Venezuela	South/Latin America	tropic
196	Vietnam	Asia & Pacific	tropic
197	Zambia	Africa	tropic
198	Zimbabwe	Africa	nontropic

[61]:
```
df_cat.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   country  199 non-null    object
 1   region   199 non-null    object
 2   climate  199 non-null    object
```

```
dtypes: object(3)
memory usage: 4.8+ KB
```

1. Dataframe 'df' terdiri dari (a) ... columns dan (b) ... rows. Dataframe 'df' mengandung data dari (c) ... 'country' yang berbeda. Dataframe 'df' mengandung data hasil observasi selama (d) .... hari terhitung sejak tanggal (e) ... sampai tanggal (f) ...

(a)
(b)
(c)
(d)
(e)
(f)

[62]:
```
df.shape
```

[62]:
```
(175840, 6)
```

[9]:
```
print('(a) ' + str(df.shape[0]))
print('(b) ' + str(df.shape[1]))
print('(c) ' + str(df.country.nunique()))
print('(d) ' + str(df.date.nunique()))
print('(e) ' + df.date.iloc[0].strftime('%d %B %Y'))
print('(f) ' + df.date.iloc[-1].strftime('%d %B %Y'))
```

```
(a) 175840
(b) 6
(c) 195
(d) 628
(e) 22 January 2020
(f) 10 October 2021
```

[64]:
```
maks_tgl = df.date
maks_tgl.max()
```

[64]:
```
Timestamp('2021-10-10 00:00:00')
```

2. Ubah beberapa data pada kolom 'country' agar tidak ada nama berbeda untuk negara yang sama dengan aturan: 'original data','replace with this data': "('St. Martin',)",'St. Martin' ' Azerbaijan','Azerbaijan' 'Cabo Verde','Cape Verde' 'Congo (Brazzaville)','Congo' 'Congo (Kinshasa)','Congo' 'North Ireland','Ireland' 'North Macedonia','Macedonia' 'occupied Palestinian territory','Palestine' 'Holy See','Vatican' 'Republic of Ireland','Ireland' 'The Bahamas','Bahamas' 'The Gambia','Gambia' 'Bahamas, The','Bahamas' 'Gambia, The','Gambia' 'Vatican City','Vatican' 'East Timor','Timor-Leste' 'West Bank and Gaza','Palestine' 'MS Zaandam','Others' 'Diamond Princess','Others'

[10]:
```
# check data
df[df.country == 'Diamond Princess']
```

[10]:

	country	province		date	confirmed	recovered	deaths
105	Diamond Princess		0	2020-01-22	0.0	0.0	0.0
384	Diamond Princess		0	2020-01-23	0.0	0.0	0.0
663	Diamond Princess		0	2020-01-24	0.0	0.0	0.0
942	Diamond Princess		0	2020-01-25	0.0	0.0	0.0
1221	Diamond Princess		0	2020-01-26	0.0	0.0	0.0
…	…	…	…	…	…	…	
173922	Diamond Princess		0	2021-10-06	712.0	0.0	13.0
174201	Diamond Princess		0	2021-10-07	712.0	0.0	13.0
174480	Diamond Princess		0	2021-10-08	712.0	0.0	13.0
174759	Diamond Princess		0	2021-10-09	712.0	0.0	13.0
175038	Diamond Princess		0	2021-10-10	712.0	0.0	13.0

[628 rows x 6 columns]

[11]:
```
# answer
replacement = {
    "('St. Martin',)": 'St. Martin',
    ' Azerbaijan': 'Azerbaijan',
    'Cabo Verde': 'Cape Verde',
    'Congo (Brazzaville)': 'Congo',
    'Congo (Kinshasa)': 'Congo',
    'North Ireland': 'Ireland',
    'North Macedonia': 'Macedonia',
    'occupied Palestinian territory': 'Palestine',
    'Holy See': 'Vatican',
    'Republic of Ireland': 'Ireland',
    'The Bahamas': 'Bahamas',
    'The Gambia': 'Gambia',
    'Bahamas, The': 'Bahamas',
    'Gambia, The': 'Gambia',
    'Vatican City': 'Vatican',
    'East Timor': 'Timor-Leste',
    'West Bank and Gaza': 'Palestine',
    'MS Zaandam': 'Others',
    'Diamond Princess': 'Others'
}

df = df.replace({'country': replacement})
df_cat = df_cat.replace({'country': replacement})
```

[12]:
```
# re-check
df[df.country == 'Diamond Princess']
```

[12]:
```
Empty DataFrame
Columns: [country, province, date, confirmed, recovered, deaths]
Index: []
```

[13]:
```
(df['country'].nunique(), df_cat['country'].nunique())
```

[13]:
```
(193, 199)
```

3. a. Beberapa 'country' pada dataframe 'df' data hariannya dibagi menjadi beberapa 'province'. Akumulasikan data 'confirmed','deaths', dan 'recovered' provinsi-provinsi ini sehingga data harian tiap negara hanya diwakili oleh 1 row dengan membuat dataframe baru 'df_new'.

b. Tambahkan kolom 'region' dan 'climate' pada 'df_new' dan isi dengan region dan climate untuk tiap negara dengan mengacu pada 'df_cat'

c. Filter 'df_new' sehingga hanya di include data tanggal 1 Maret 2020 - 13 Desember 2020. Drop semua row pada 'df_new' yang data 'confirmed'-nya di bawah 100. Drop semua row yang data 'region' atau 'climate'-nya NaN pada 'df_new'.

d. Buat line plot berdasarkan dataframe 'df_new' dengan data 'date' sebagai x dan data 'confirmed' sebagai y, di mana tiap garis mewakili data total (bukan rata-rata) 1 region.

e. Buat line plot berdasarkan dataframe 'df_new' dengan data 'date' sebagai x dan data 'confirmed' sebagai y, di mana tiap garis mewakili data total (bukan rata-rata) 1 kelompok iklim.

[14]:
```
# (a)
df_new = df.groupby(['country', 'date'], as_index=False).sum()
df_new
```

[14]:

	country	date	confirmed	recovered	deaths
0	Afghanistan	2020-01-22	0.0	0.0	0.0
1	Afghanistan	2020-01-23	0.0	0.0	0.0
2	Afghanistan	2020-01-24	0.0	0.0	0.0
3	Afghanistan	2020-01-25	0.0	0.0	0.0
4	Afghanistan	2020-01-26	0.0	0.0	0.0
…	…	…	…	…	…
121199	Zimbabwe	2021-10-06	131434.0	0.0	4630.0
121200	Zimbabwe	2021-10-07	131523.0	0.0	4631.0
121201	Zimbabwe	2021-10-08	131705.0	0.0	4634.0
121202	Zimbabwe	2021-10-09	131762.0	0.0	4636.0
121203	Zimbabwe	2021-10-10	131796.0	0.0	4637.0

[121204 rows x 5 columns]

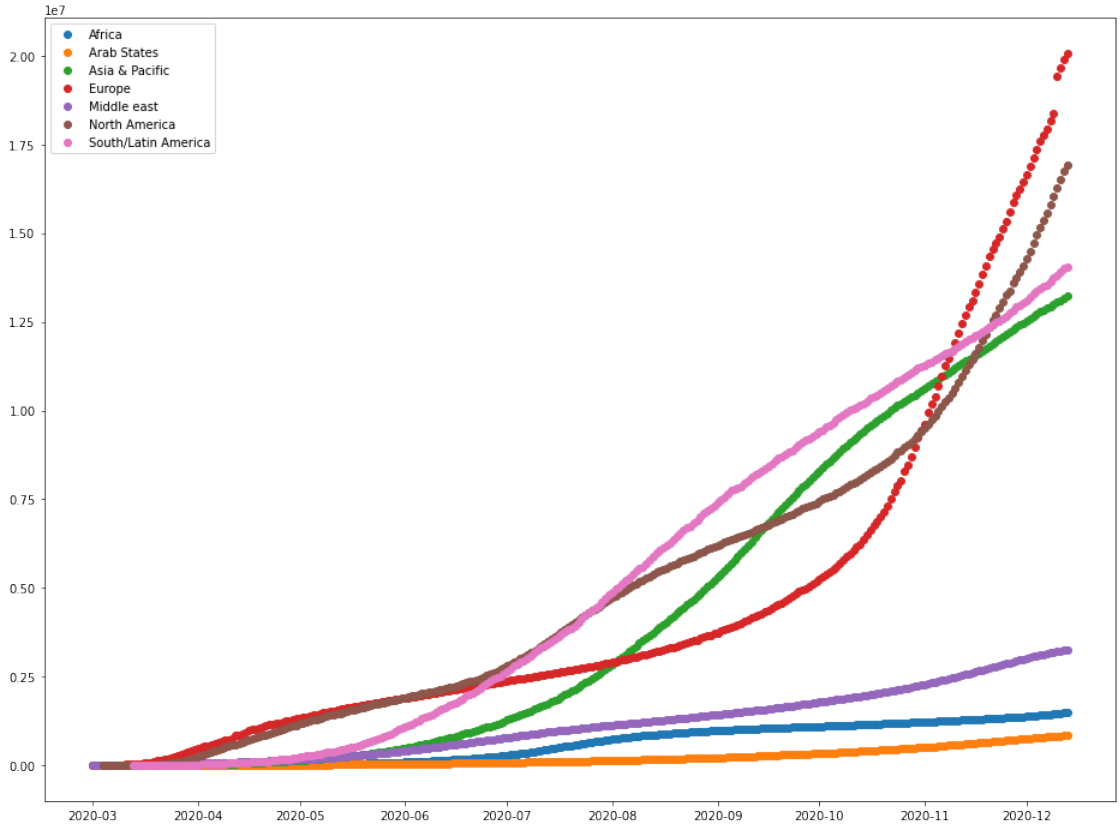
[15]:
```
# (b)
df_new = df_new.merge(df_cat, how='outer', on='country')
df_new
```

[15]:
```
                country       date  …              region    climate
0           Afghanistan 2020-01-22  …      Asia & Pacific  nontropic
1           Afghanistan 2020-01-23  …      Asia & Pacific  nontropic
2           Afghanistan 2020-01-24  …      Asia & Pacific  nontropic
3           Afghanistan 2020-01-25  …      Asia & Pacific  nontropic
4           Afghanistan 2020-01-26  …      Asia & Pacific  nontropic
…                   …        …  …                 …        …
121226  Saint Barthelemy        NaT  …  South/Latin America     tropic
121227       South Korea        NaT  …      Asia & Pacific  nontropic
121228        St. Martin        NaT  …  South/Latin America     tropic
121229            Taiwan        NaT  …      Asia & Pacific  nontropic
121230                UK        NaT  …              Europe  nontropic

[121231 rows x 7 columns]
```

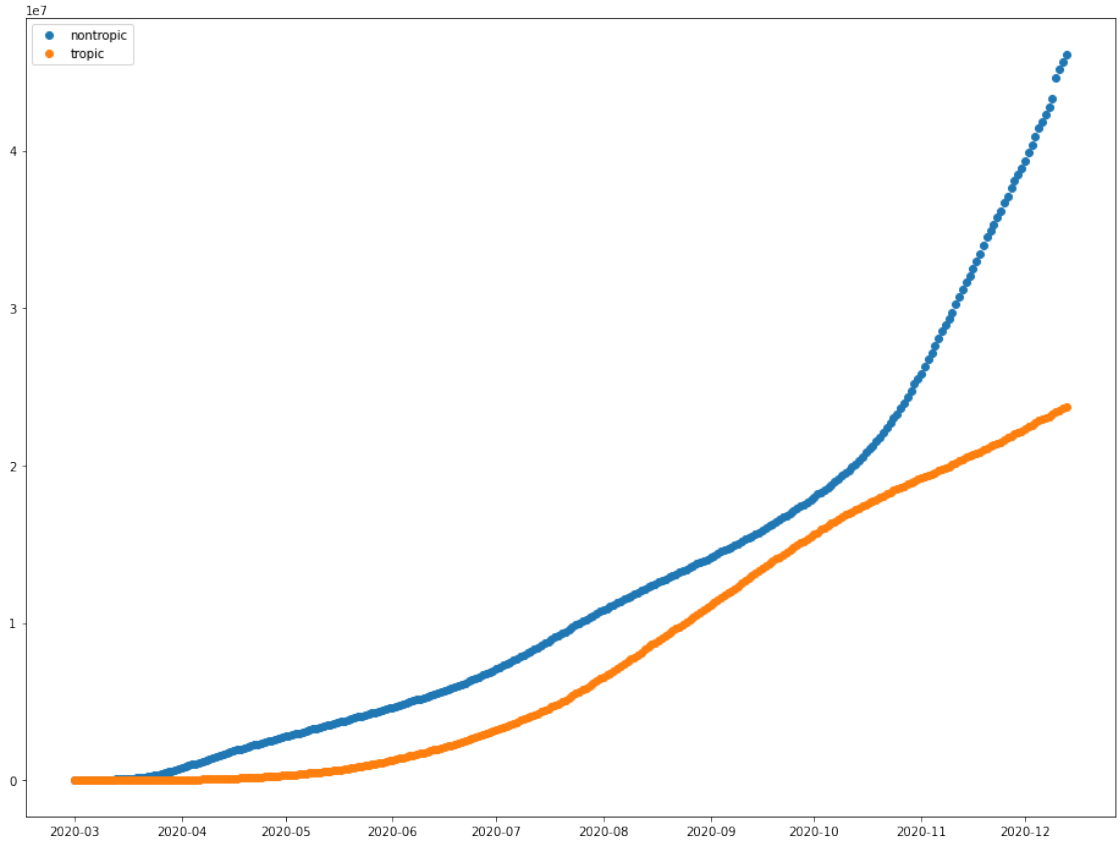
[16]:
```
# (c)
df_new = df_new[(df_new.date >= '2020-03-01') & (df_new.date <= '2020-12-13')]
df_new = df_new[~(df_new.confirmed < 100)]
df_new = df_new.dropna(subset=['region', 'climate'])
df_new
```

[16]:
```
            country       date  confirmed  …  deaths          region
climate
66      Afghanistan 2020-03-28      106.0  …     2.0  Asia & Pacific
nontropic
67      Afghanistan 2020-03-29      114.0  …     4.0  Asia & Pacific
nontropic
68      Afghanistan 2020-03-30      114.0  …     4.0  Asia & Pacific
nontropic
69      Afghanistan 2020-03-31      166.0  …     4.0  Asia & Pacific
nontropic
70      Afghanistan 2020-04-01      192.0  …     4.0  Asia & Pacific
nontropic
…               …        …         …  …     …              …
…
120898     Zimbabwe 2020-12-09    11007.0  …   304.0          Africa
nontropic
120899     Zimbabwe 2020-12-10    11081.0  …   305.0          Africa
nontropic
120900     Zimbabwe 2020-12-11    11162.0  …   306.0          Africa
nontropic
120901     Zimbabwe 2020-12-12    11219.0  …   307.0          Africa
nontropic
120902     Zimbabwe 2020-12-13    11246.0  …   307.0          Africa
nontropic

[40729 rows x 7 columns]
```

[17]:
```
# (d)
df_new_3d = df_new.groupby(['region', 'date'], as_index=False).sum()

plt.figure(figsize=(16,12))
for region in df_new_3d.region.unique():
    df_temp = df_new_3d[df_new_3d.region == region]
    plt.plot_date(x=df_temp['date'], y=df_temp['confirmed'], label=region)

plt.legend()
plt.show()
```



[18]:
```
# (e)
df_new_3e = df_new.groupby(['climate', 'date'], as_index=False).sum()

plt.figure(figsize=(16,12))
for climate in df_new_3e.climate.unique():
    df_temp = df_new_3e[df_new_3e.climate == climate]
    plt.plot_date(x=df_temp['date'], y=df_temp['confirmed'], label=climate)

plt.legend()
plt.show()
```



4.a. Buat dataframe 'df_last' yang hanya mengandung data tanggal terakhir dari 'df_new', di mana tiap row menunjukkan data 'confirmed', 'deaths', 'recovered', 'region', dan 'climate' untuk 1 negara.

b. Berdasarkan df_last, identifikasi 10 negara dengan data 'deaths' tertinggi. Buat barplotnya.

c. Buat beberapa kolom baru pada 'df_last':

Kolom 'active_case' yang merupakan hasil perhitungan 'confirmed' dikurangi 'recovered' dan 'deaths'.

Kolom 'active_case_%' yang merupakan hasil perhitungan 'active_case' dibagi 'confirmed' dikali 100.

Kolom 'deaths_%' yang merupakan hasil perhitungan 'deaths' dibagi 'confirmed' dikali 100.

Kolom 'recovered_%' yang merupakan hasil perhitungan 'recovered' dibagi 'confirmed' dikali 100.

d. Buat scatter matrix berdassarkan 'df_last' untuk kolom 'active_case_%', 'deaths_%', dan 'recovered_%' (bedakan warna scatter plot berdasarkan region).

e. Lakukan hal yang sama dengan membedakan warna scatter plot berdasarkan iklim.

[19]:
```
df_new.date.iloc[-1]
```

[19]:
```
Timestamp('2020-12-13 00:00:00')
```

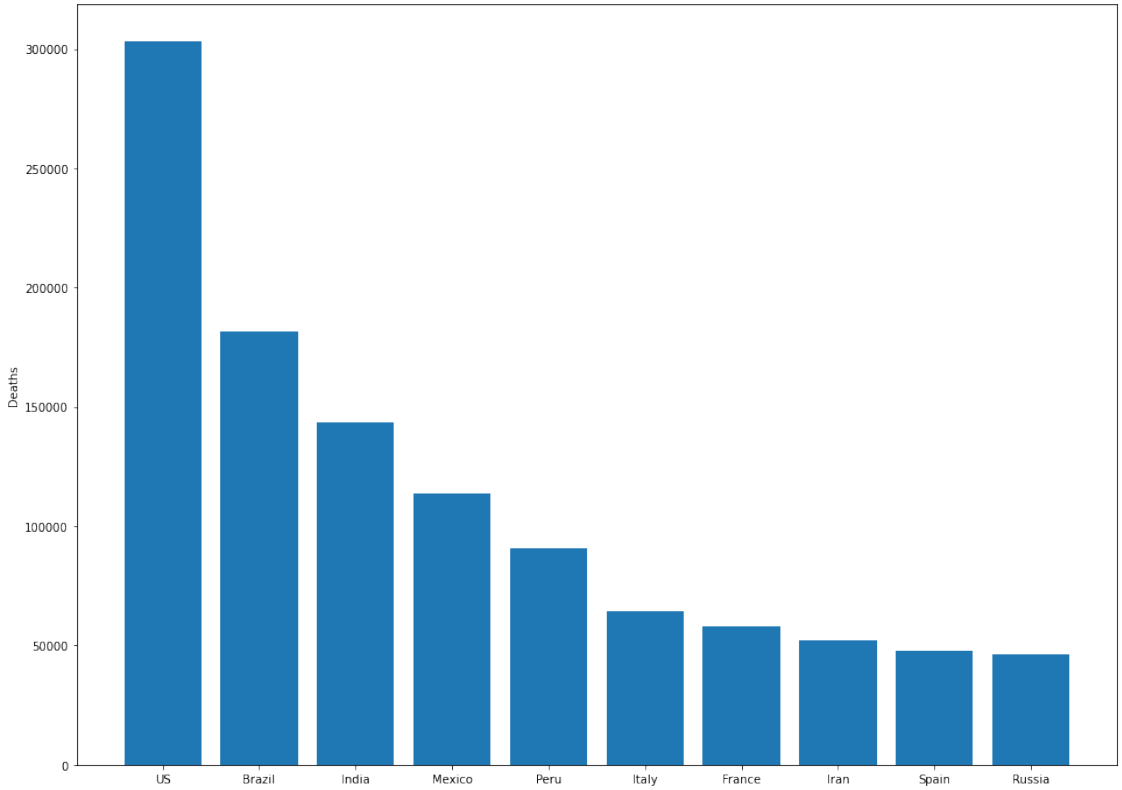
[20]:
```
# (a)
df_last = df_new[df_new.date == '2020-12-13']
df_last = df_last.drop(columns=['date'])
df_last
```

[20]:

	country	confirmed	…	region	climate
326	Afghanistan	48952.0	…	Asia & Pacific	nontropic
954	Albania	48530.0	…	Europe	nontropic
1582	Algeria	92102.0	…	Arab States	nontropic
2210	Andorra	7338.0	…	Europe	nontropic
2838	Angola	16188.0	…	Africa	tropic
…	…	…	…	…	…
116506	Uzbekistan	74956.0	…	Asia & Pacific	nontropic
118390	Venezuela	107786.0	…	South/Latin America	tropic
119018	Vietnam	1397.0	…	Asia & Pacific	tropic
120274	Zambia	18274.0	…	Africa	tropic
120902	Zimbabwe	11246.0	…	Africa	nontropic

[164 rows x 6 columns]

[21]:
```
# (b)
df_last_4b = df_last.sort_values(by=['deaths'], ascending=False).head(10)

x_coords = np.arange(len(df_last_4b))
plt.figure(figsize=(16,12))
plt.bar(x_coords, df_last_4b.deaths, tick_label=df_last_4b.country)
plt.ylabel('Deaths')
plt.show()
```



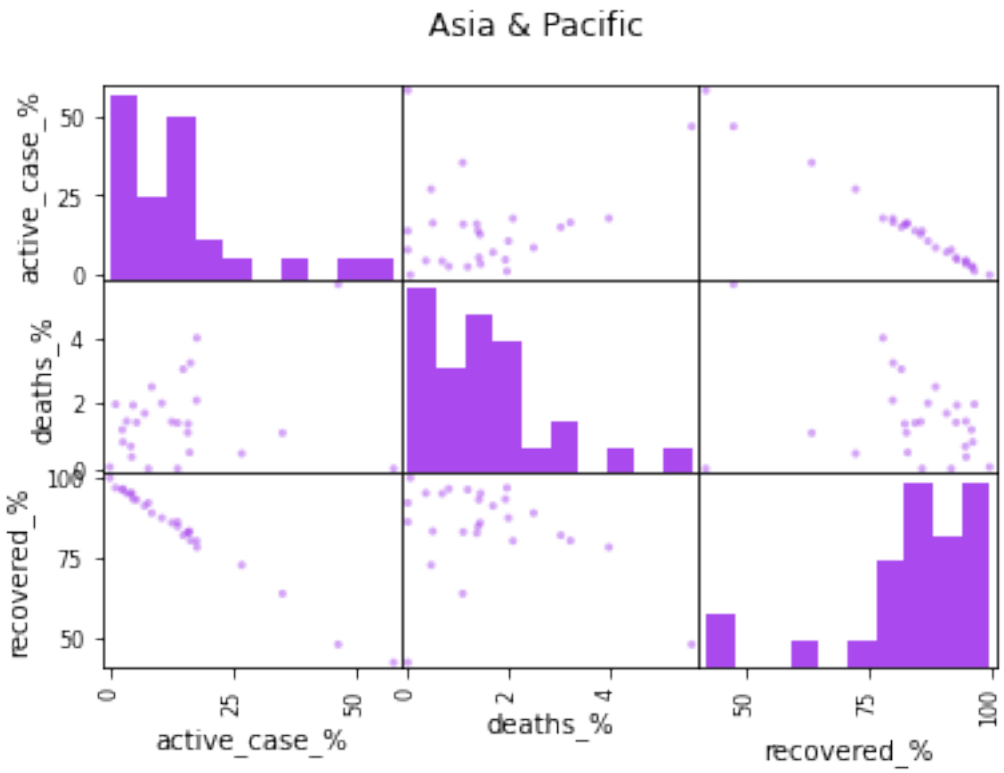
[22]:
```
# (c)
df_last['active_case'] = df_last.confirmed - df_last.recovered - df_last.deaths
df_last['active_case_%'] = (df_last.active_case / df_last.confirmed) * 100
df_last['deaths_%'] = (df_last.deaths / df_last.confirmed) * 100
df_last['recovered_%'] = (df_last.recovered / df_last.confirmed) * 100
df_last
```

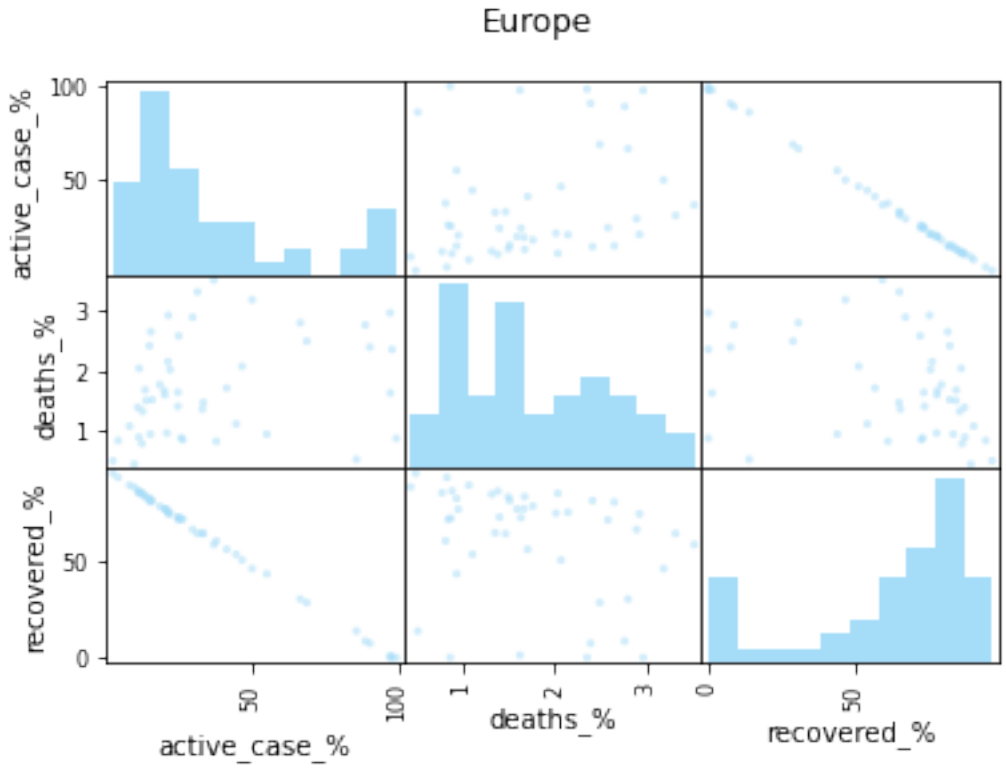
[22]:

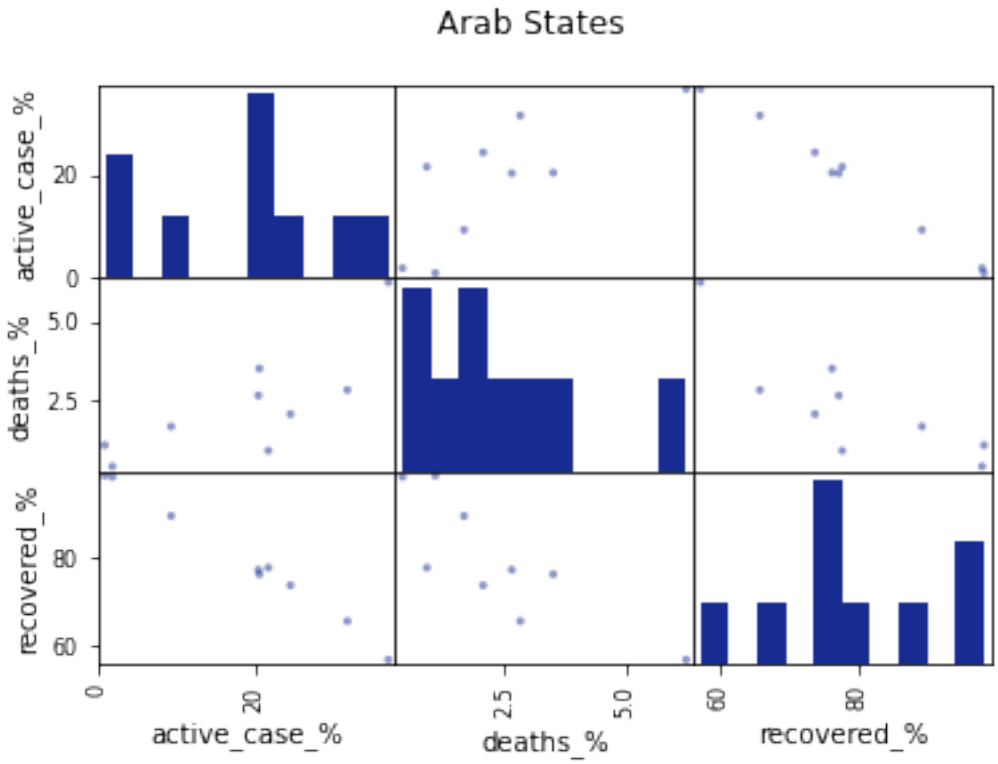
	country	confirmed	recovered	…	active_case_%	deaths_% recovered_%
326	Afghanistan	48952.0	38250.0	…	17.858310	4.003922 78.137768
954	Albania	48530.0	24820.0	…	46.789615	2.066763 51.143623
1582	Algeria	92102.0	60457.0	…	31.540032	2.818614 65.641354
2210	Andorra	7338.0	6629.0	…	8.585446	1.076588 90.337967
2838	Angola	16188.0	8898.0	…	42.741537	2.291821 54.966642
…	…	…	…	…	…	… …
116506	Uzbekistan	74956.0	72243.0	…	2.802978	0.816479

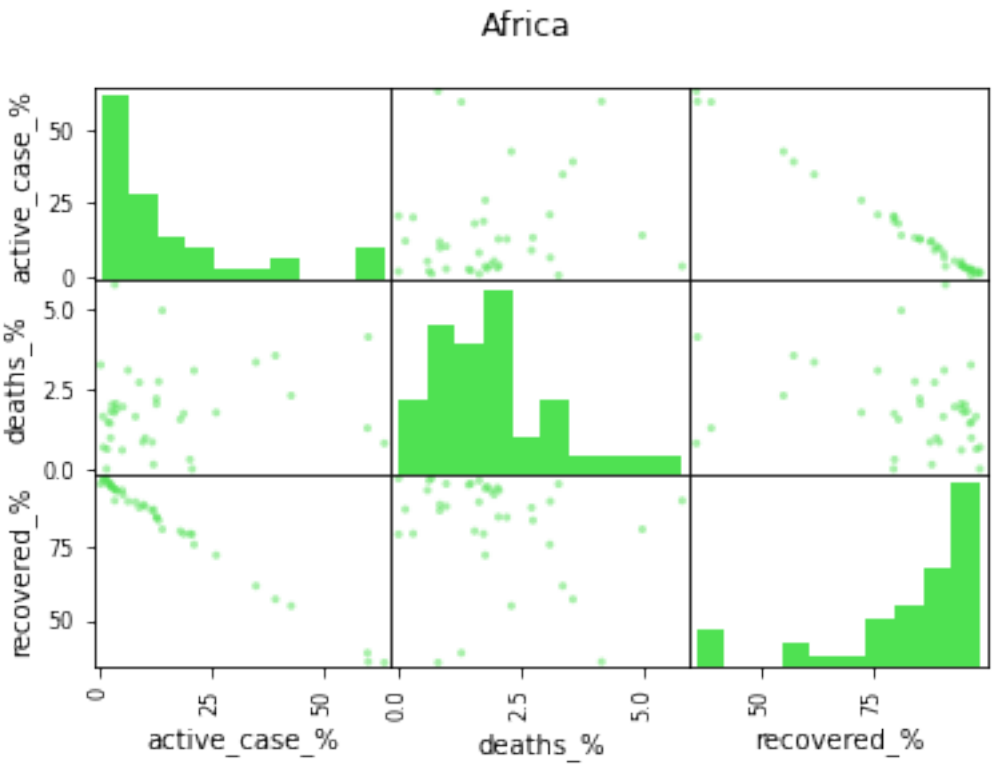
```
96.380543
118390    Venezuela   107786.0   102289.0  …       4.214833  0.885087
94.900080
119018      Vietnam     1397.0     1241.0  …       8.661417  2.505369
88.833214
120274       Zambia    18274.0    17388.0  …       2.840101  2.008318
95.151581
120902     Zimbabwe    11246.0     9451.0  …      13.231371  2.729860
84.038769

[164 rows x 10 columns]
```

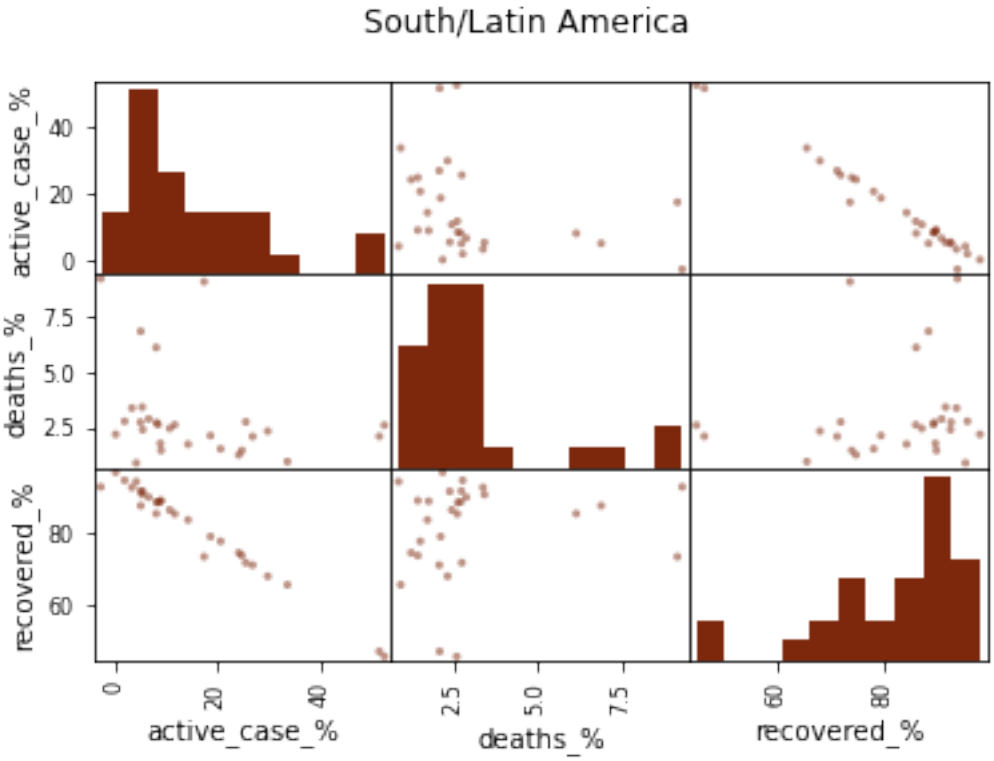
[23]:
```
# (d)
import random

def random_color():
    return "#{}{}{}{}{}{}".format(*(random.choice("0123456789abcdef") for _ in␣
 ↪range(6)))

df_last_4d = df_last[['region'] + list(df_last.columns[7:10])]

colors = [random_color() for _ in range(len(df_last.region.unique()))]
for count, region in enumerate(df_last.region.unique()):
    df_temp = df_last_4d[df_last_4d.region == region]
    pd.plotting.scatter_matrix(df_temp, c=colors[count], hist_kwds={'color':
 ↪colors[count]})
    plt.suptitle(region)
```

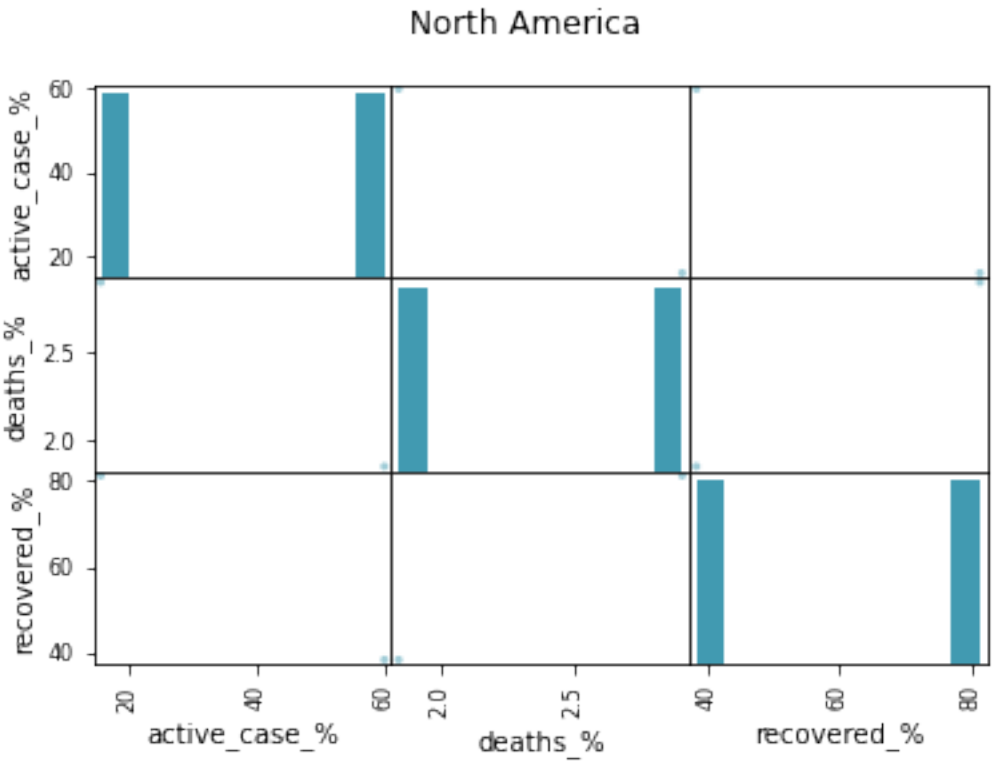

Asia & Pacific
active_case_%
deaths_%
recovered_%
0
25
50
0
2
4
50
75
100

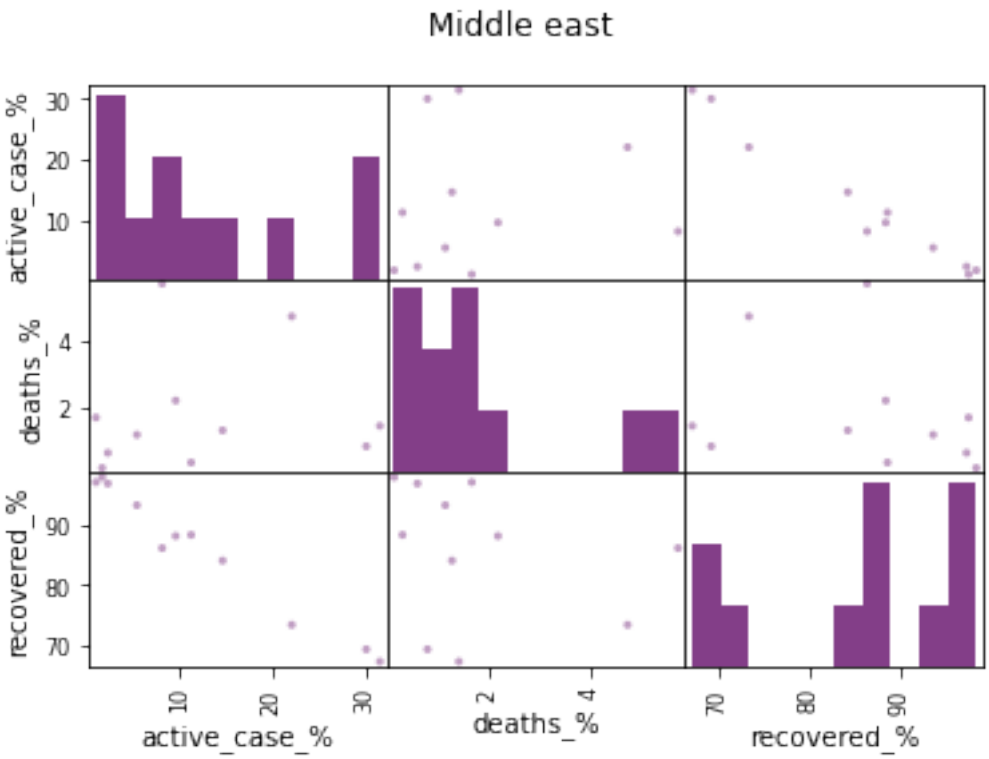

Europe
active_case_%
deaths_%
recovered_%
0
50
100
1
2
3
0
50


Arab States
active_case_%
deaths_%
recovered_%
0
20
5.0
2.5
60
80

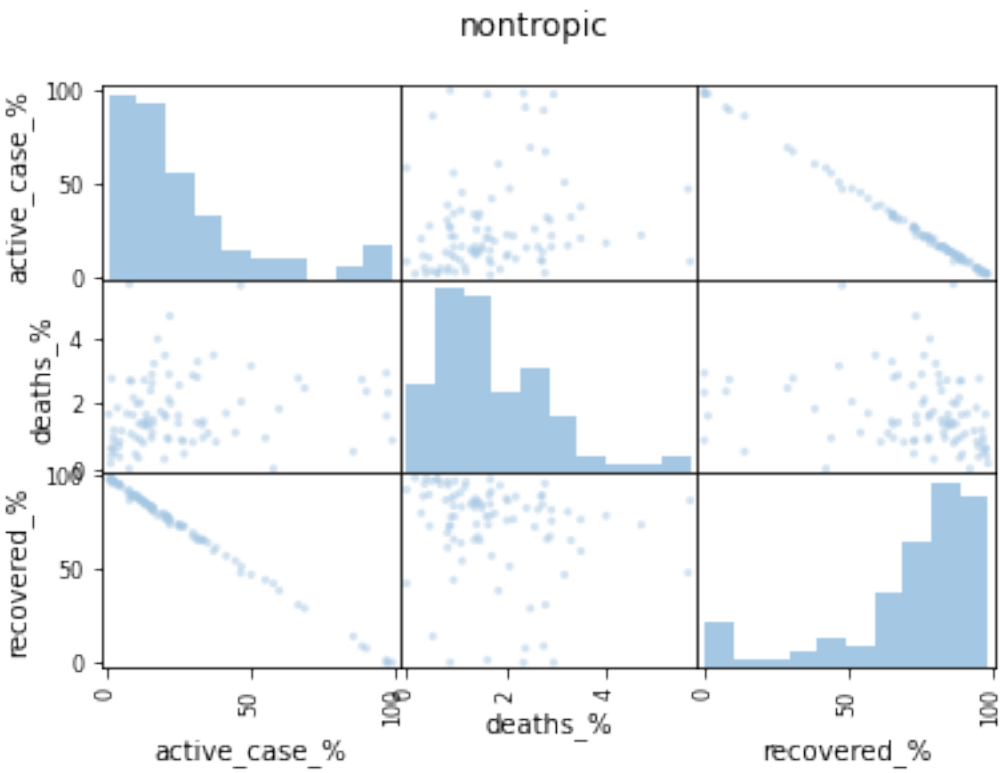

Africa
active_case_%
deaths_%
recovered_%
0
25
50
0.0
2.5
5.0
50
75

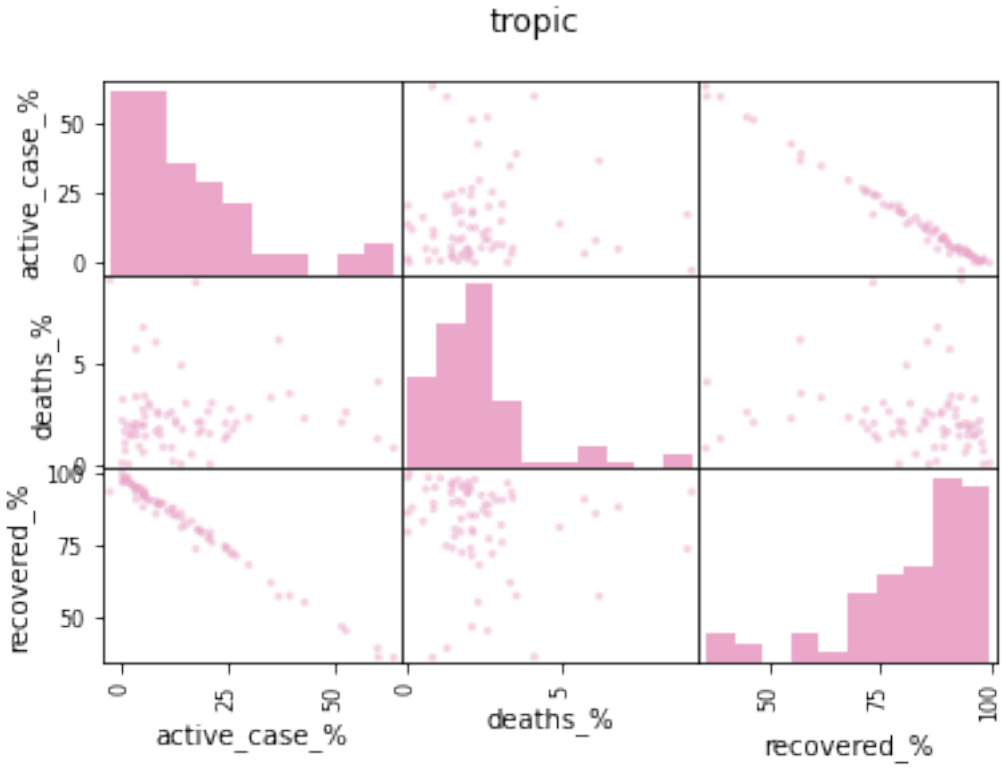

South/Latin America
active_case_%
deaths_%
recovered_%
0
20
40
2.5
5.0
7.5
60
80


North America
active_case_%
deaths_%
recovered_%
60
40
20
2.5
2.0
80
60
40
20
40
60
2.0
2.5
40
60
80



[24]:
```
# (e)
df_last_4e = df_last[['climate'] + list(df_last.columns[7:10])]

colors = [random_color() for _ in range(len(df_last.climate.unique()))]
for count, climate in enumerate(df_last.climate.unique()):
    df_temp = df_last_4e[df_last_4e.climate == climate]
    pd.plotting.scatter_matrix(df_temp, c=colors[count], hist_kwds={'color':
 ↪colors[count]})
    plt.suptitle(climate)
```


nontropic
active_case_%
deaths_%
recovered_%


tropic
active_case_%
deaths_%
recovered_%

5. a. Dengan asumsi syarat melakukan tes Pearson terpenuhi, lakukan tes Pearson untuk menguji adanya hubungan linear antara data 'active_case_%', 'deaths_%', dan 'recovered_%'. Buatlah heatmapnya.

b. Carilah p-value dari koefisiensi korelasi yang telah didapatkan. Tuliskan kesimpulannya.

[25]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

df_last_5 = df_last[list(df_last.columns[7:10])]
df_last_5 = StandardScaler().fit_transform(df_last_5)
df_last_5 = pd.DataFrame(df_last_5, index=df_last.index, columns=list(df_last.
 ↪columns[7:10]))
df_last_5
```

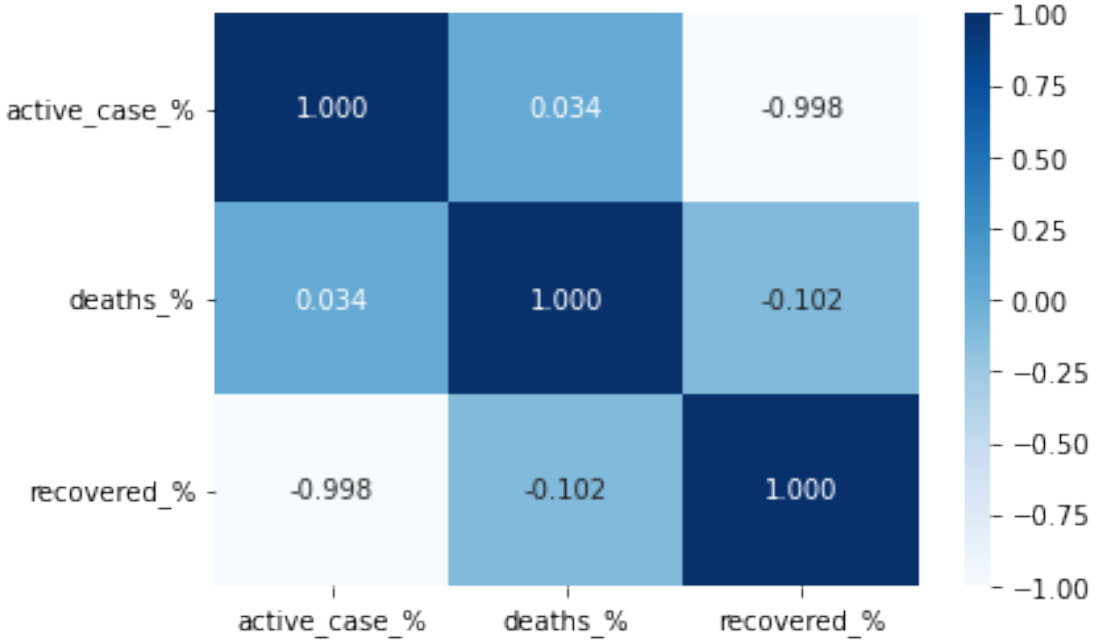
[25]:

	active_case_%	deaths_%	recovered_%
326	-0.154613	1.336931	0.062484
954	1.176864	0.039071	-1.174032
1582	0.475047	0.542797	-0.509937
2210	-0.581369	-0.624328	0.621336
2838	0.990563	0.189856	-0.998912
…	…	…	…
116506	-0.847490	-0.798595	0.898128
118390	-0.782514	-0.752629	0.830312
119018	-0.577873	0.332929	0.552408
120274	-0.845781	-0.000086	0.841833
120902	-0.367554	0.483333	0.332790

[164 rows x 3 columns]

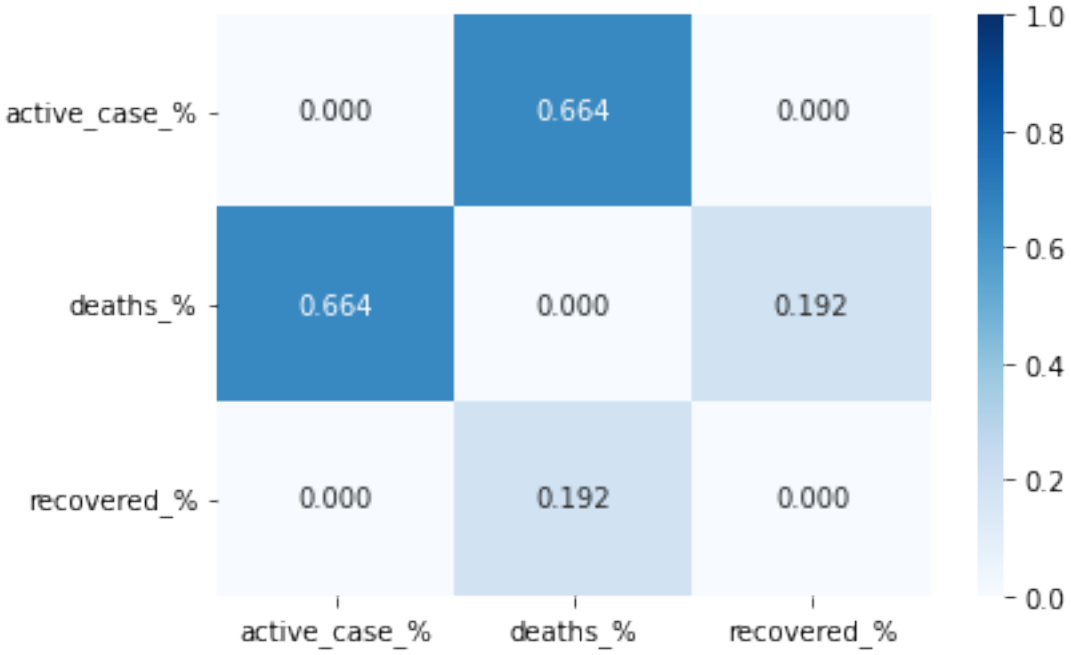
[26]:
```
# (a)
from scipy import stats
import seaborn as sns

row1 = []
row2 = []
row3 = []
for i in range(3):
    row1.append(stats.pearsonr(df_last_5[df_last_5.columns[0]],␣
 ↪df_last_5[df_last_5.columns[i]])[0])
    row2.append(stats.pearsonr(df_last_5[df_last_5.columns[1]],␣
 ↪df_last_5[df_last_5.columns[i]])[0])
    row3.append(stats.pearsonr(df_last_5[df_last_5.columns[2]],␣
 ↪df_last_5[df_last_5.columns[i]])[0])
```

```
heatmap_data = np.array([row1,
                         row2,
                         row3])

ax = sns.heatmap(heatmap_data, vmin=-1.0, vmax=1.0, cmap=plt.cm.Blues,
                 annot=True, fmt='.3f',
                 xticklabels=df_last_5.columns, yticklabels=df_last_5.columns)
plt.yticks(rotation=0)
plt.show()
```



[27]:
```
# (b)
row1 = []
row2 = []
row3 = []

for i in range(3):
    row1.append(stats.pearsonr(df_last_5[df_last_5.columns[0]],␣
 ↪df_last_5[df_last_5.columns[i]])[1])
    row2.append(stats.pearsonr(df_last_5[df_last_5.columns[1]],␣
 ↪df_last_5[df_last_5.columns[i]])[1])
    row3.append(stats.pearsonr(df_last_5[df_last_5.columns[2]],␣
 ↪df_last_5[df_last_5.columns[i]])[1])

heatmap_data = np.array([row1,
                         row2,
```

```
                        row3])

ax = sns.heatmap(heatmap_data, vmin=0.0, vmax=1.0, cmap=plt.cm.Blues,
                 annot=True, fmt='.3f',
                 xticklabels=df_last_5.columns, yticklabels=df_last_5.columns)
plt.yticks(rotation=0)
plt.show()
```



6. Dengan asumsi syarat melakukan tes ANOVA terpenuhi, lakukan ANOVA untuk menguji adanya hubungan antara region dan data 'active_case_%', 'deaths_%', dan 'recovered_%' (3 TES ANOVA YANG BERBEDA) pada 'df_last'. Tuliskan kesimpulan dari hasil tesnya.

[28]:
```
from sklearn import preprocessing

encoder = preprocessing.LabelEncoder() # bisa diganti dengan one hot enconding␣
↪karena label lebih dari 2, sangat disarankan
region = encoder.fit_transform(df_last['region'])

print(encoder.classes_)
print(region)
```

```
['Africa' 'Arab States' 'Asia & Pacific' 'Europe' 'Middle east'
 'North America' 'South/Latin America']
[2 3 1 3 0 6 6 3 2 3 2 6 1 2 6 3 3 6 0 2 6 3 0 6 2 3 0 2 0 2 0 5 0 0 0 6 6
 0 6 3 6 3 3 1 6 6 4 6 0 0 3 0 0 3 3 0 0 3 3 0 3 6 0 0 6 6 6 3 3 2 2 4 4 3
```

```
 3 3 6 2 4 2 0 3 4 2 3 4 0 4 3 3 3 3 0 0 2 2 0 3 1 0 6 3 3 2 3 1 0 0 2 3 2
 6 0 0 3 4 2 1 6 2 6 6 2 3 3 4 3 3 0 6 3 4 0 3 0 0 2 3 3 1 0 3 2 1 6 3 3 2
 0 2 0 6 1 3 5 0 3 4 6 2 6 2 0 0]
```

[29]:
```
fvalue, pvalue = stats.f_oneway(region, df_last['active_case_%'])
print(fvalue, pvalue)
## nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan region␣
 ↪terhadap kasus aktif
```

```
118.20359936783886 1.065335258178832e-23
```

[30]:
```
fvalue, pvalue = stats.f_oneway(region, df_last['deaths_%'])
print(fvalue, pvalue)
# nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan region␣
 ↪terhadap kematian
```

```
10.08823292225845 0.0016348352053110035
```

[31]:
```
fvalue, pvalue = stats.f_oneway(region, df_last['recovered_%'])
print(fvalue, pvalue)
# nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan region␣
 ↪terhadap penyembuhan
```

```
1863.929911034642 6.984108187339506e-137
```

7. Dengan asumsi syarat melakukan independent t-test terpenuhi, lakukan independent t-test untuk menguji adanya hubungan antara kelompok iklim dan data 'active_case_%', 'deaths_%', dan 'recovered_%' (3 INDEPENDENT T-TEST YANG BERBEDA) pada 'df_last'. Tuliskan kesimpulan dari hasil tesnya.

[32]:
```
encoder = preprocessing.LabelEncoder()
climate = encoder.fit_transform(df_last['climate'])

print(encoder.classes_)
print(climate)
```

```
['nontropic' 'tropic']
[0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 1
 1 1 0 1 0 0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0
 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0
 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0
 1 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0]
```

[33]:
```
tvalue, pvalue = stats.ttest_ind(climate, df_last['active_case_%'])
print(tvalue, pvalue)
# nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan iklim␣
 ↪terhadap kasus aktif
```

```
-12.205831718509563 1.7781276297443802e-28
```

[34]:
```
tvalue, pvalue = stats.ttest_ind(climate, df_last['deaths_%'])
print(tvalue, pvalue)
# nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan iklim␣
 ↪terhadap kematian
```

```
-12.738754747236579 1.921382588003052e-30
```

[35]:
```
tvalue, pvalue = stats.ttest_ind(climate, df_last['recovered_%'])
print(tvalue, pvalue)
# nilai p sangat kecil (<0.05) sehingga terdapat pengaruh signifikan iklim␣
 ↪terhadap penyembuhan
```

```
-44.6306938131577 6.633833287310479e-141
```

8. Buat kolom 'safety' pada 'df_last'. Apabila data suatu negara memenuhi 'active_case_%' < 10, 'deaths_%' < 5, 'recovered_%' > 85 labeli negara ini dengan 'safe' pada kolom 'safety'. Apabila tidak memenuhi kriteria, labeli dengan 'not_safe'. Berapa negara yang masuk kategori 'safe'? Berapa negara yang masuk kategori 'not_safe'?

[36]:
```
safety = []

for row in df_last.to_dict(orient="records"):
    if ((row['active_case_%'] < 10) and (row['deaths_%'] < 5) and␣
 ↪(row['recovered_%'] > 85)):
        result = "safe"
    else:
        result = "not_safe"
    safety.append(result)

df_last['safety'] = safety
df_last
```

[36]:

	country	confirmed	recovered	…	deaths_%	recovered_%	safety
326	Afghanistan	48952.0	38250.0	…	4.003922	78.137768	not_safe
954	Albania	48530.0	24820.0	…	2.066763	51.143623	not_safe
1582	Algeria	92102.0	60457.0	…	2.818614	65.641354	not_safe
2210	Andorra	7338.0	6629.0	…	1.076588	90.337967	safe
2838	Angola	16188.0	8898.0	…	2.291821	54.966642	not_safe
…	…	…	…	…	…	…	
116506	Uzbekistan	74956.0	72243.0	…	0.816479	96.380543	safe
118390	Venezuela	107786.0	102289.0	…	0.885087	94.900080	safe
119018	Vietnam	1397.0	1241.0	…	2.505369	88.833214	safe
120274	Zambia	18274.0	17388.0	…	2.008318	95.151581	safe
120902	Zimbabwe	11246.0	9451.0	…	2.729860	84.038769	not_safe

[164 rows x 11 columns]

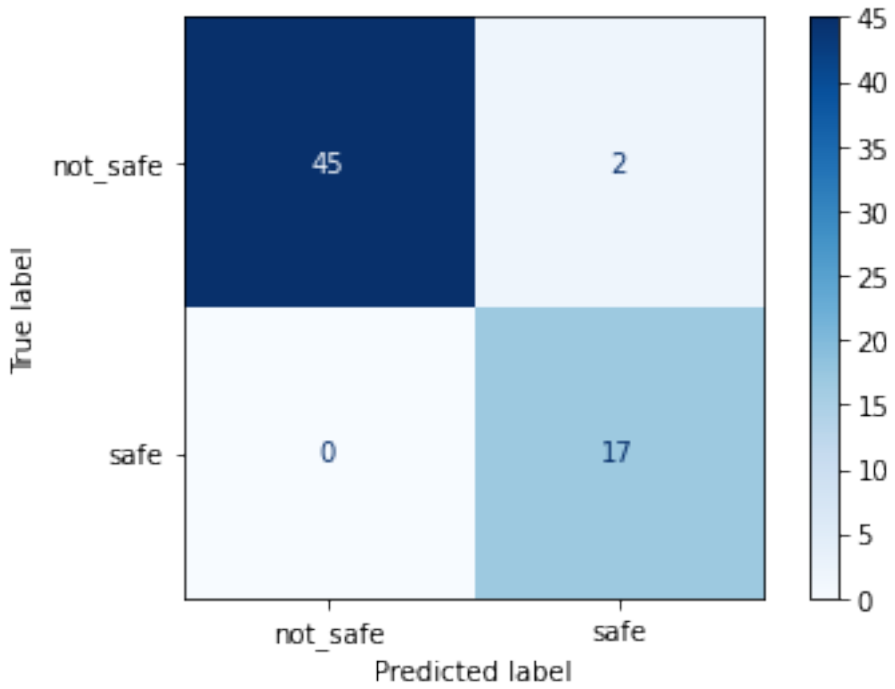
9. a. Buat DataFrame baru dengan yang mengeksklusi data Indonesia, US, Brazil, New Zealand, Singapore dari 'df_last'. Buatlah classifier model untuk memprediksi kategori 'safety' pada DataFrame yang baru dibuat. Tampilkan nilai performance dari model yang Anda buat.

b. Gunakan model classifier yang Anda buat untuk melabeli kategory 'safety' pada data Indonesia, US, Brazil, New Zealand, Singapore dari 'df_last'. Apa label 'safety' yang diprediksi oleh model Anda untuk masing-masing negara tersebut? Apakah hasil prediksi model Anda sesuai dengan kriteria safety pada nomor 8? Apabila tidak, faktor apa yang mempengaruhinya?

[37]:
```
# (a)
exclude = (
    (df_last.country == 'Indonesia') |
    (df_last.country == 'US') |
    (df_last.country == 'Brazil') |
    (df_last.country == 'New Zealand')
)

df_last_9a = df_last[~exclude]
df_last_9b = df_last[exclude]
```

[38]:
```
from sklearn import preprocessing
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, plot_confusion_matrix

x = df_last_9a[['active_case_%', 'deaths_%', 'recovered_%']].to_numpy()

encoder = preprocessing.LabelEncoder()
y = encoder.fit_transform(df_last_9a.safety)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4,␣
 ↪random_state=42)

clf = tree.DecisionTreeClassifier().fit(x_train, y_train)
y_pred = clf.predict(x_test)

plot_confusion_matrix(clf, x_test, y_test, display_labels=encoder.classes_,␣
 ↪cmap=plt.cm.Blues)
plt.show()
print(classification_report(y_test, y_pred, target_names=encoder.classes_))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



	precision	recall	f1-score	support
not_safe	1.00	0.96	0.98	47
safe	0.89	1.00	0.94	17
accuracy			0.97	64
macro avg	0.95	0.98	0.96	64
weighted avg	0.97	0.97	0.97	64

[39]:
```
# (b)
x_9b = df_last_9b[['active_case_%', 'deaths_%', 'recovered_%']].to_numpy()

y_pred_9b = clf.predict(x_9b)
y_pred_9b = y_pred_9b.tolist()
y_pred_9b = ['safe' if (float(x)==1.0) else 'not_safe' for x in y_pred_9b]
y_pred_9b

df_last_9b = df_last_9b.assign(predict=y_pred_9b)
df_last_9b[['country', 'safety', 'predict']]
```

[39]:

	country	safety	predict
14770	Brazil	safe	safe
48682	Indonesia	not_safe	not_safe
77570	New Zealand	safe	safe

```
112738           US  not_safe  not_safe
```

10. Buatlah model regresi untuk data 'active_case_%' di US pada 'df_last'. Plot model regresi ini bersama data aslinya dalam satu graph. Hitung nilai performance dari model regresi yang Anda buat.

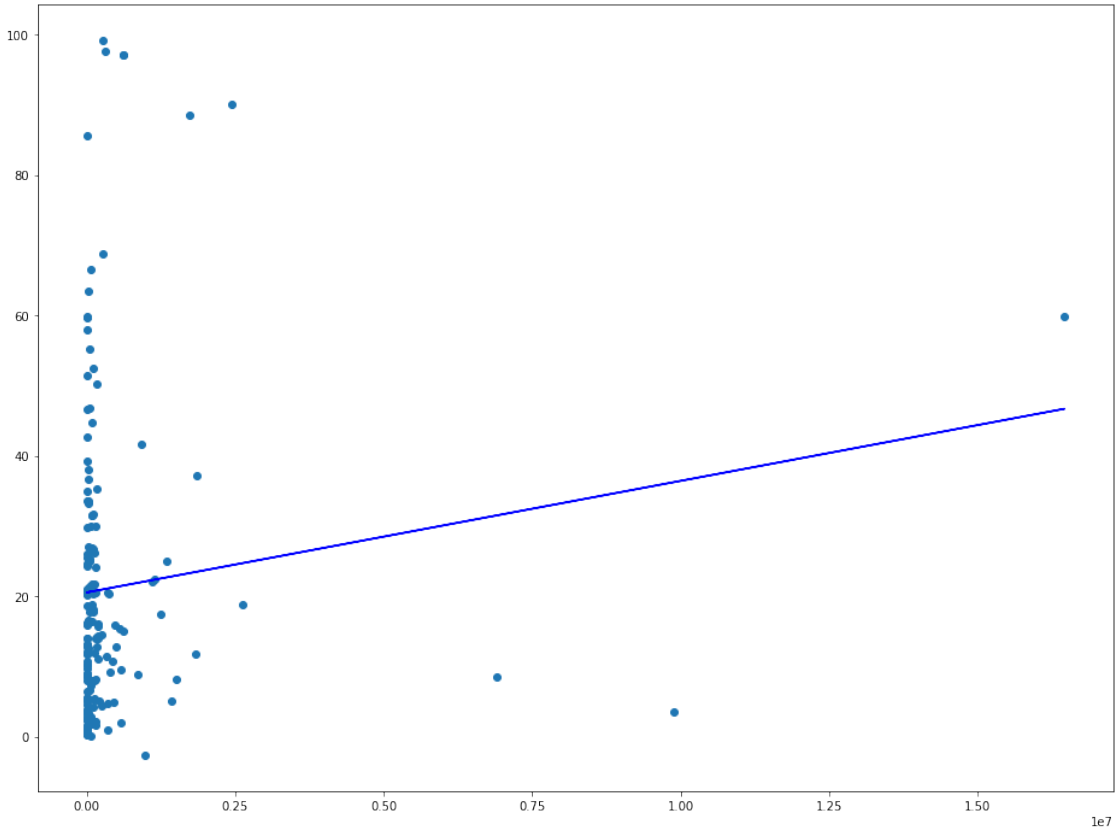
[40]:
```
df_last[df_last.country == 'US']
```

[40]:
```
        country   confirmed  recovered  …  deaths_%  recovered_%    safety
112738      US  16463227.0  6298082.0  …  1.843278    38.255453  not_safe

[1 rows x 11 columns]
```

[41]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

x = df_last[['confirmed']].to_numpy()
y = df_last[['active_case_%']].to_numpy()

LR = LinearRegression().fit(x, y)
y_pred = LR.predict(x)

US_predictor = df_last[df_last.country == 'US'][['confirmed']].to_numpy()

print('Prediksi active_case_% di US: ' + str(LR.predict(US_predictor)[0]))
print('Nilai mean-squared error model: ' + str(mean_squared_error(y, y_pred)))
print('Nilai R2 model: ' + str(r2_score(y, y_pred)))
```

```
Prediksi active_case_% di US: [46.6920573]
Nilai mean-squared error model: 465.5703209375214
Nilai R2 model: 0.013910195165635963
```

[42]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(16,12))
plt.scatter(x,y)
plt.plot(x,y_pred,'b')
# plt.xlim(0,0.25*1e6)
plt.show()
```



[ ]:
```
# Karena datanya terlalu menyebar dan terdapat outlier maka perlu dilakukan␣
↪normalisasi dan imputasi sebelum dilakukan regresi
```

11. a. Buatlah clustering model dengan jumlah cluster sebanyak 5 (cluster 0-4) untuk array 'Z'. Sebutkan prediksi nomor cluster berdasarkan data 'active_case_%', 'deaths_%', dan 'recovered_%' pada 'df_last' untuk negara:

i. Indonesia

ii. Singapore

iii. US

iv. Italy

v. Iran

b. Buat plot 3D dari clustering model yang Anda buat.

[43]:
```
# (a)
from sklearn.cluster import KMeans
```

```
from mpl_toolkits.mplot3d import Axes3D

Z = df_last.loc[:,['active_case_%', 'deaths_%', 'recovered_%']].values

cluster = KMeans(n_clusters=5, random_state=42)
cluster.fit(Z)

df_last['cluster_number'] = cluster.labels_
df_last
```

[43]:

	country	confirmed	…	safety	cluster_number
326	Afghanistan	48952.0	…	not_safe	4
954	Albania	48530.0	…	not_safe	1
1582	Algeria	92102.0	…	not_safe	2
2210	Andorra	7338.0	…	safe	0
2838	Angola	16188.0	…	not_safe	2
…	…	…	…	…	…
116506	Uzbekistan	74956.0	…	safe	0
118390	Venezuela	107786.0	…	safe	0
119018	Vietnam	1397.0	…	safe	0
120274	Zambia	18274.0	…	safe	0
120902	Zimbabwe	11246.0	…	not_safe	4

[164 rows x 12 columns]

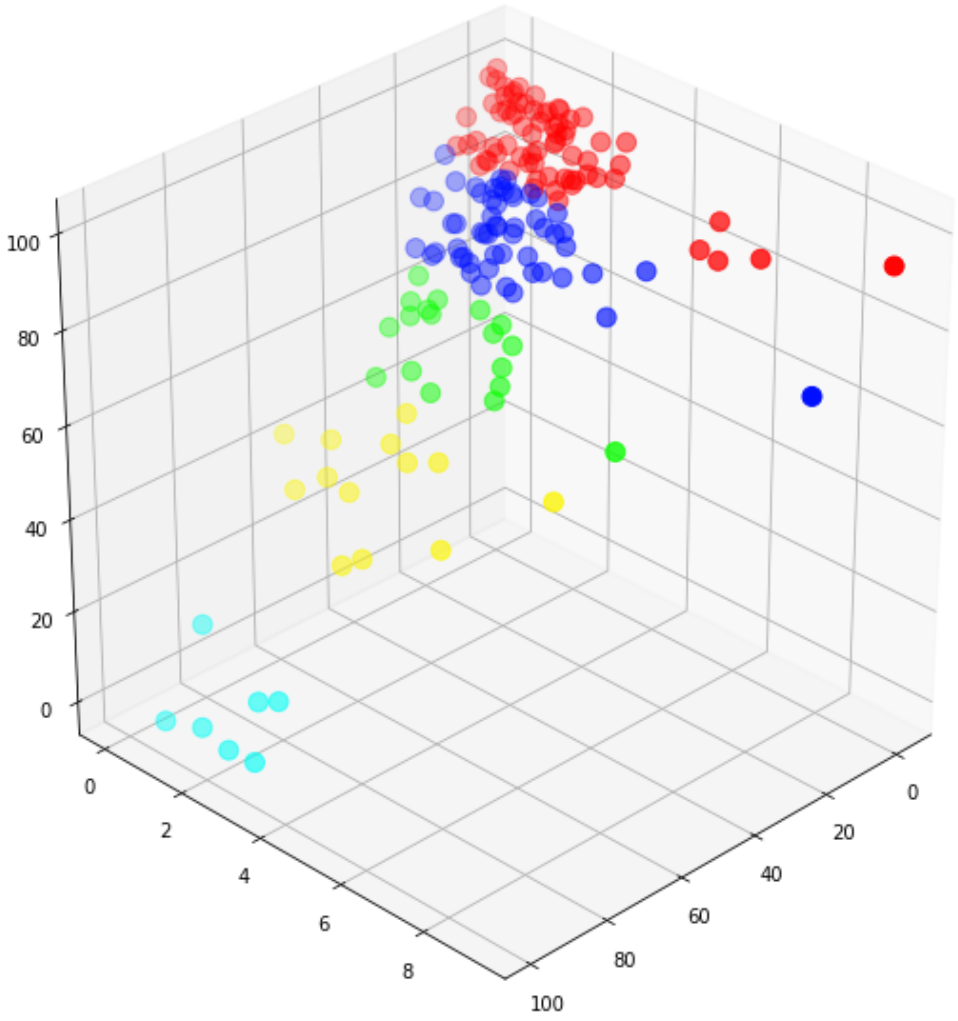
[44]:
```
filter = (
    (df_last['country'] == 'Indonesia') |
    (df_last['country'] == 'Singapore') |
    (df_last['country'] == 'US') |
    (df_last['country'] == 'Italy') |
    (df_last['country'] == 'Iran')
)

df_last[filter][['country', 'active_case_%', 'deaths_%', 'recovered_%',␣
↪'cluster_number']]
```

[44]:

	country	active_case_%	deaths_%	recovered_%	cluster_number
48682	Indonesia	15.079635	3.046033	81.874332	4
49310	Iran	21.998540	4.709687	73.291773	4
51822	Italy	37.209228	3.499462	59.291310	2
97666	Singapore	0.142318	0.049726	99.807956	0
112738	US	59.901270	1.843278	38.255453	1

[45]:
```
# (b)
fig = plt.figure(figsize=(8,8))
ax = Axes3D(fig)
```

```
ax.scatter(Z[:,0], Z[:,1], Z[:,2], vmin=0, vmax=6, cmap='hsv', c=cluster.
 ↪labels_, s=100)
ax.view_init(azim=45)
plt.show()
```



12. (Optional) Tampilkan grafik-grafik yang Anda buat dalam bentuk dashboard menggunakan Dash by Plotly.

[ ]:
```
%%shell
jupyter nbconvert --to html 'library session pandas covid.ipynb'
```

[46]:
```
```