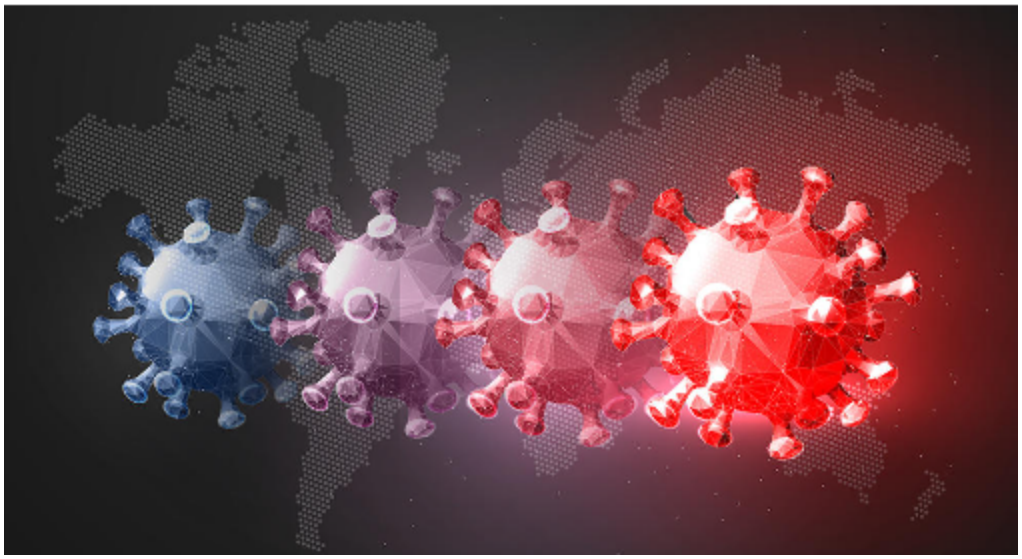


Analysis objective

1. Total number of deaths and cases of COVID-19 worldwide
2. Total number of deaths and cases of COVID-19 in each continent
3. Top 05 countries with the highest deaths and cases of COVID
4. Bottom 05 countries with the lowest deaths and cases of COVID
5. Daily cumulative trend of COVID-19 cases and deaths
6. Total number of deaths and cases by country per year/month
7. Annual mortality rate
8. Monthly infection rate
9. Monthly average total cases (infection) and total deaths (mortality)
10. COVID-19 vaccinations trend

> *The dataset is made up of COVID-19 records for the years 2020/2021/2022/2023(download date 09/01/2023 = to the last day of records in the dataset).*
> ***LINK TO COVID DATASET***

In [1]:
```
import pandas as pd
import numpy as np
data = pd.read_csv(r"C:\Users\pc\Downloads\owid-covid-data (1).csv")
data.head(2)
```

Out[1]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	NaN	NaN
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	NaN	NaN

2 rows × 67 columns

PART A (data cleaning)

1. Get familiar with the data set
2. Check the data type of the columns
3. Select the columns useful for analysis
4. Check for null values
5. Change some cells' value
6. Cell filling
7. Deletion of unnecessary and redundant lines

2) Display the table dimensions and columns names

In [2]:
```
data.columns
data.shape
# 24 8856 records and 67 columns
```

Out[2]:
```
(248856, 67)
```

3) Check the data type of each column

In [3]:
```
data.dtypes
# the date column type is "object" we must parse it to "date"
```

Out[3]:
```
iso_code                                 object
continent                                object
location                                 object
date                                     object
total_cases                             float64
                                          ...
population                              float64
excess_mortality_cumulative_absolute    float64
excess_mortality_cumulative             float64
excess_mortality                        float64
excess_mortality_cumulative_per_million float64
Length: 67, dtype: object
```

4) Parse the date column into "datetime" column type

In [4]:
```
data["date"] = pd.to_datetime(data["date"])
data["date"].dtypes
```

Out[4]:
```
dtype('<M8[ns]')
```

5) Choose our columns

In [5]:
```
data = data.iloc[:,[0,1,2,3,4,5,7,8,34,35,36,62]]
data.head(2)
#data.dtypes
```

Out[5]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	p
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	NaN	NaN	
1	AFG	Asia	Afghanistan	2020-	5.0	0.0	NaN	NaN	NaN	

02-25

- **_observation_**

> For each record, for a location for a **given date** we have **total_cases**, **new_cases**, **total_deaths** *etc*.
>
> The columns **total_cases** and **total_deaths** are ***cumulative columns*** so as a day increase we have the ***cumulative sum***
> of all the values of the ***previous days***;
>
> **new_deaths**, **new_cases** are ***not cumulative columns***
> **Population** is the same value for a country regardless of the day.

6) Null values

In [6]:
```
data.isnull().sum()
# we have columns that contain null cells
# depending on our analysis we will see if we delete them or we fill them
```

Out[6]:
```
iso_code                      0
continent                 13986
location                      0
date                          0
total_cases               14148
new_cases                 14477
total_deaths              33657
new_deaths                33757
total_vaccinations       178534
people_vaccinated        181603
people_fully_vaccinated  184291
population                 1068
dtype: int64
```

7) Null cells in the "continent" column

In [7]:
```
data[data["continent"].isna()].head(3)
```

Out[7]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations
1051	OWID_AFR	NaN	Africa	2020-02-13	NaN	0.0	NaN	0.0	NaN
1052	OWID_AFR	NaN	Africa	2020-02-14	1.0	1.0	NaN	0.0	NaN
1053	OWID_AFR	NaN	Africa	2020-02-15	1.0	0.0	NaN	0.0	NaN

In [8]:
```
data["continent"].isna().sum()
```

Out[8]:
```
13986
```

In [9]:
```
data["location"][data["continent"].isna()].value_counts()
```

Out[9]:
```
Asia                    1084
High income             1084
Lower middle income     1084
```

```
North America           1084
Upper middle income     1084
World                   1084
Europe                  1083
European Union          1083
Oceania                 1081
International           1068
Africa                  1062
South America           1053
Low income              1052
Name: location, dtype: int64
```

- **_observation_**
- For all the null cells in the continent column the values in the location column is a group location (not a country):
  - If we had a country as a corresponding value would have filled the corresponding null cell in the continent column by the
    country continent, but we do not have that case
  - we will leave the nulls value this way, the null cells are justified

8) I want the location column to only contain countries

In [10]:
```
data=data[~((data["continent"].isnull()) & (data["location"].isin(["High income","Low in
```

9) Let's verify that the deletion has been carried out

In [11]:
```
data[(data["continent"].isnull()) & (data["location"].isin(["High income","Low income","
#we have no records, we are ok
```

Out[11]:

iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	people

In [12]:
```
data.isnull().sum()  # due to the cleaning we did above, continent and location no longe
```

Out[12]:
```
iso_code                      0
continent                     0
location                      0
date                          0
total_cases               14141
new_cases                 14477
total_deaths              33465
new_deaths                33744
total_vaccinations       173493
people_vaccinated        176562
people_fully_vaccinated  179070
population                    0
dtype: int64
```

10) Duplicates

In [13]:
```
data[data.duplicated()] # no duplicates
```

Out[13]:

iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	people

11) Total cases & New_cases, Total deaths & New deaths, People vaccinated & People fully vaccinated

In [14]:
```
data[ data[ "new_cases"] > data[ "total_cases"]].sort_values("date", ascending = True, n
# we have no records we are good
```

Out[14]:

iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	people

In [15]:
```
data[(data[ "new_deaths"] > data[ "total_deaths"])]
# we have no records we are good
```

Out[15]:

iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	people

In [16]:
```
data[(data[ "people_fully_vaccinated"] > data[ "people_vaccinated"])]
# we have no records we are good
```

Out[16]:

iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_vaccinations	people

- **_observation_**

> **_Total cases should be $>=$ new*cases***:
> The purpose is to verify if this condition is true, if it isn't, try to understand why?
>
> **_Total deaths should be $>=$ new*deaths***;
>
> **_People_vaccinated should be $>=$ people_fully*vaccinated***.

12) Null values in quantitative columns

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
iso_code                      0
continent                     0
location                      0
date                          0
total_cases               14141
new_cases                 14477
total_deaths              33465
new_deaths                33744
total_vaccinations       173493
people_vaccinated        176562
people_fully_vaccinated  179070
population                    0
dtype: int64
```

- **_observation_**

> We have null values in almost all the ***quantitative columns*** except ***population***
>
> For the moment we will not fill the cells with any value
> Depending on the analatycal query, will fill the cells or not...let just keep it in mind

PART B (Analysis)

> This part consists of answering analytical questions and visualising them

> We should keep in mind that the quantitative columns:***Total deaths*** and ***Total cases*** have values that are cumulated daily for each location
> The ***max value of a quantitative column should actually corresepond to the last day of our dataset for a given location***
> ex .. ***total deaths*** in ***20/09/2020*** should be >= to all the ***total deaths*** of the previous days.
> The columns **_new*deaths*** and **_new*cases*** are not cumulative columns
>
> For a ***location (country)*** first record of deaths or cases, **_total*cases*** is equal to **_new*cases***, but as days go by the **_total*cases*** column becomes **> =** to the **_new*case*** column
>
> ***Total cases*** is a daily cumulated column and **_new*cases*** is a daily not cumulated column both for each location

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

13) Total number of deaths and cases of COVID worldwide up to **09/01/2023**

In [19]:
```
data.groupby("location").max().loc[:,["total_cases","total_deaths"]].sum()
```

Out[19]:
```
total_cases     664703388.0
total_deaths      6709331.0
dtype: float64
```

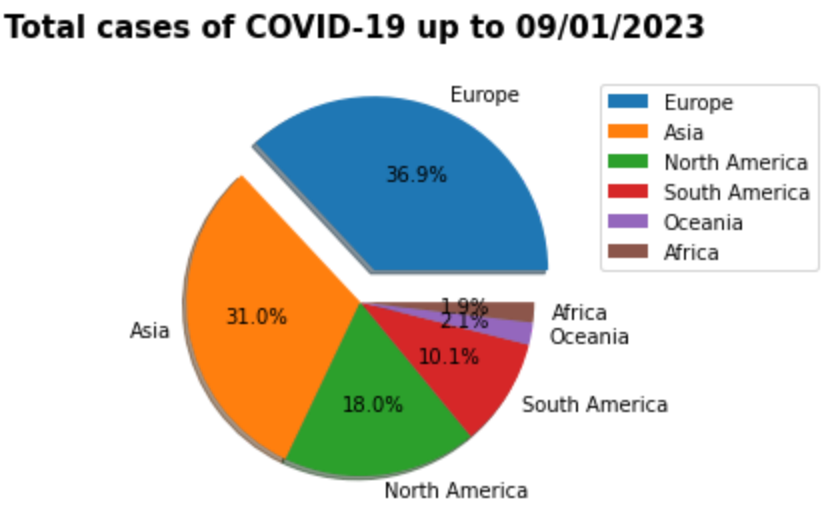
14) Total number of deaths and cases of COVID in each continent up to **09/01/2023**

In [20]:
```
df = data.groupby(["continent","iso_code","location"]).max().loc[:,["total_cases","total
df.groupby(["continent"]).sum().loc[:,["total_cases","total_deaths"]].sort_values(["tota
```

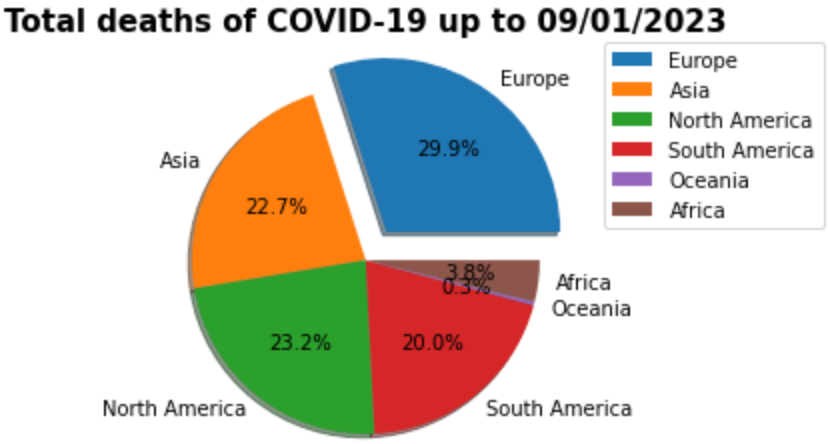
Out[20]:

continent	total_cases	total_deaths
Europe	245591654.0	2007405.0
Asia	205944055.0	1522037.0
North America	119827464.0	1556673.0
South America	67107304.0	1342857.0
Oceania	13756680.0	22571.0
Africa	12476231.0	257788.0

In [21]:
```
df = df.groupby(["continent"]).sum().loc[:,["total_cases","total_deaths"]].sort_values([

myexplode = [0.2, 0, 0, 0,0,0]
plt.pie(df["total_cases"], labels = df["continent"],autopct='%1.1f%%', explode = myexplo

plt.title("Total cases of COVID-19 up to 09/01/2023 \n", fontsize=15, fontweight="bold")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```



In [22]:
```
myexplode = [0.2, 0, 0, 0,0,0]
plt.pie(df["total_deaths"], labels = df["continent"], autopct='%1.1f%%',explode = myexpl

plt.title("Total deaths of COVID-19 up to 09/01/2023",fontsize=15, fontweight="bold")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```



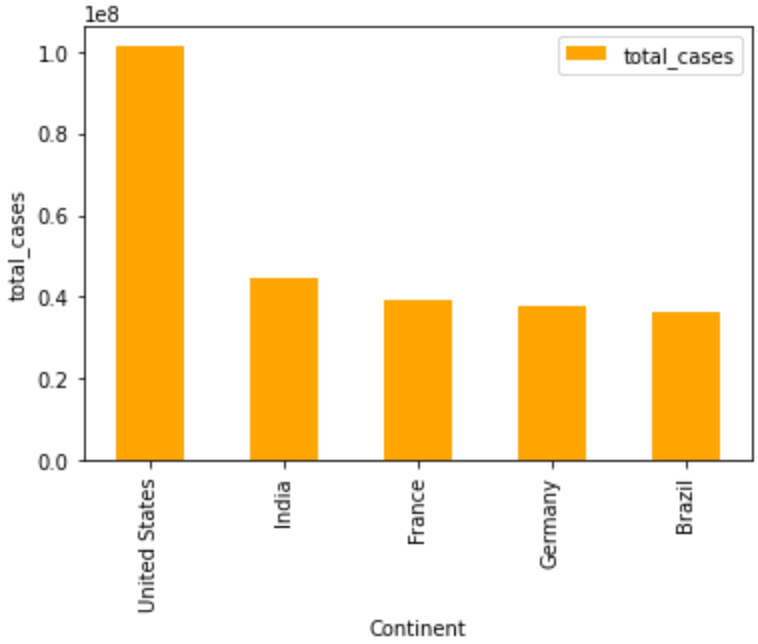
15) Top 05 countries with the highest deaths and cases of COVID up to **_09/01/2023_**

In [23]:
```
data.groupby("location").max().loc[:,["total_cases","total_deaths"]].sort_values(["total
```

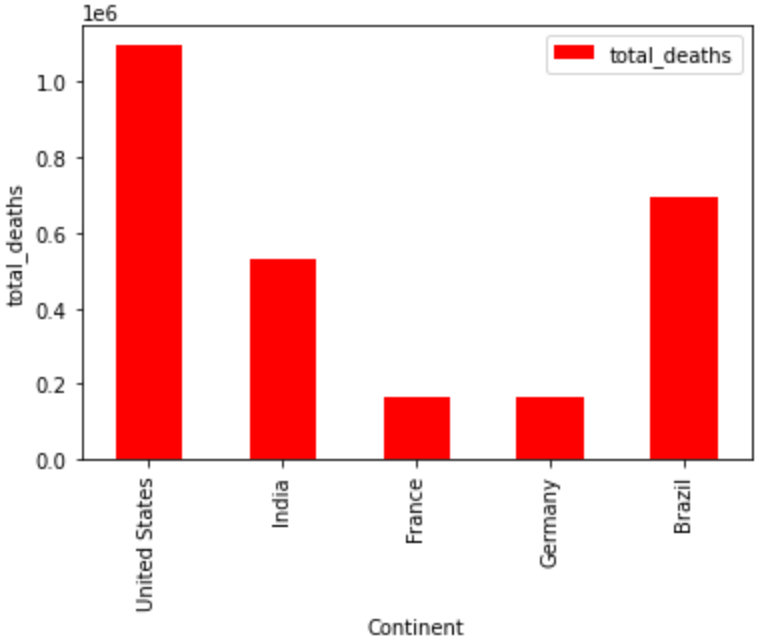
Out[23]:

location	total_cases	total_deaths
United States	101285347.0	1096751.0
India	44681439.0	530722.0
France	39449416.0	163059.0
Germany	37540072.0	162975.0
Brazil	36477214.0	694779.0

In [24]:
```
df_country = data.groupby("location").max().loc[:,["total_cases","total_deaths"]].sort_v
df_country.plot(kind="bar",y=["total_cases"], color = "orange")
plt.title("Top 05 countries with the highest cases of COVID-19 up to 09/01/2023 \n", fon
plt.ylabel("total_cases")
plt.xlabel("Continent")
plt.show()
```



In [25]:
```
df_country.plot(kind="bar",y=["total_deaths"], color = "red")
plt.title("Top 05 countries with the highest deaths of COVID-19 up to 09/01/2023 \n", fo
plt.ylabel("total_deaths")
plt.xlabel("Continent")
plt.show()
```



16) Bottom 05 countries with the lowest deaths and cases of COVID up to **_09/01/2023_**

In [26]:
```
data.groupby("location").max().loc[:,["total_cases","total_deaths"]].sort_values(["total
```

Out[26]:

location	total_cases	total_deaths
North Korea	1.0	6.0
Vatican	29.0	NaN

Montserrat	1403.0	8.0
Falkland Islands	1930.0	NaN
Saint Helena	2166.0	NaN

17) Let us insert the year and month column **_09/01/2023_**

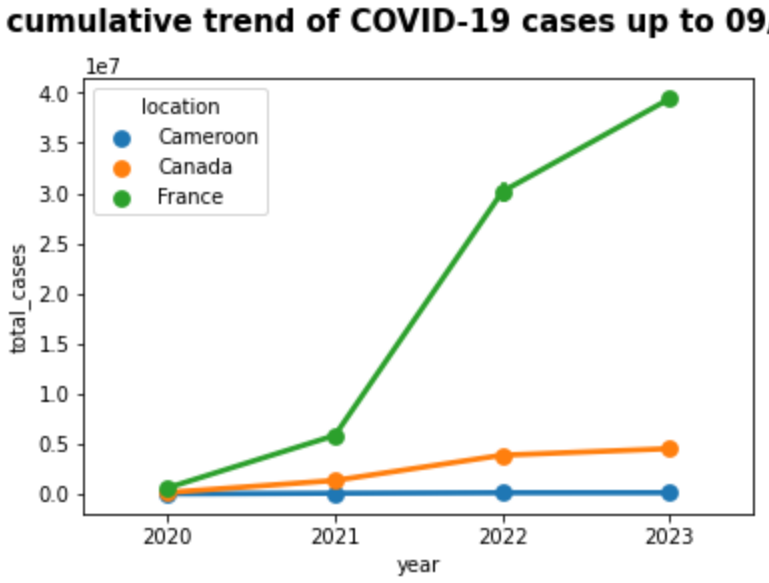
In [27]:
```
data.insert(3,"month",data["date"].dt.month)
data.insert(4,"year",data["date"].dt.year)
data.head(3)
```

Out[27]:

	iso_code	continent	location	month	year	date	total_cases	new_cases	total_deaths	new_deaths	total_
0	AFG	Asia	Afghanistan	2	2020	2020-02-24	5.0	5.0	NaN	NaN	
1	AFG	Asia	Afghanistan	2	2020	2020-02-25	5.0	0.0	NaN	NaN	
2	AFG	Asia	Afghanistan	2	2020	2020-02-26	5.0	0.0	NaN	NaN	

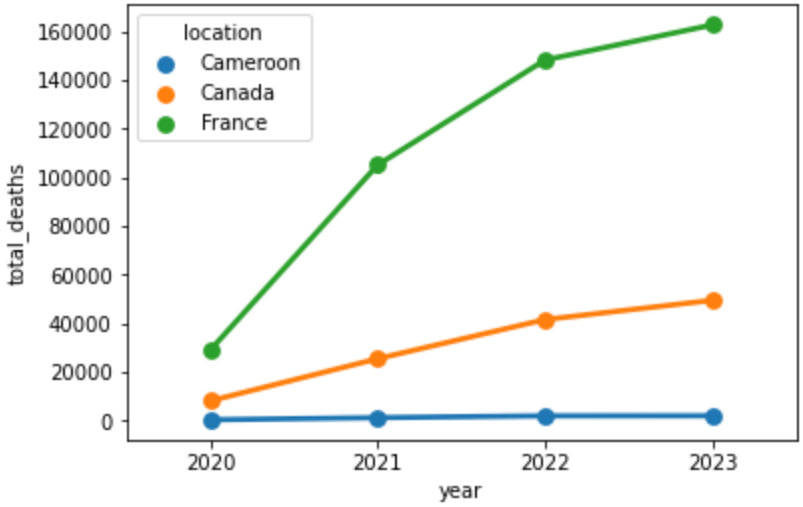
18) Daily cumulative trend of COVID-19 cases and deaths up to **_09/01/2023_**

In [28]:
```
data_cum = data[data["location"].isin(["France","Canada","Cameroon"])]  # I decided to c
plt.subplot(111)
sns.pointplot(x='year', y='total_cases', data = data_cum, hue= "location")

plt.title("Daily cumulative trend of COVID-19 cases up to 09/01/2023 \n", fontsize=15, f
plt.show()
```



In [29]:
```
plt.subplot(111)
sns.pointplot(x='year', y='total_deaths', data = data_cum, hue= "location")

plt.title("Daily cumulative trend of COVID-19 deaths up to 09/01/2023 \n", fontsize=15,
plt.show()
```

Daily cumulative trend of COVID-19 deaths up to 09/01/2023

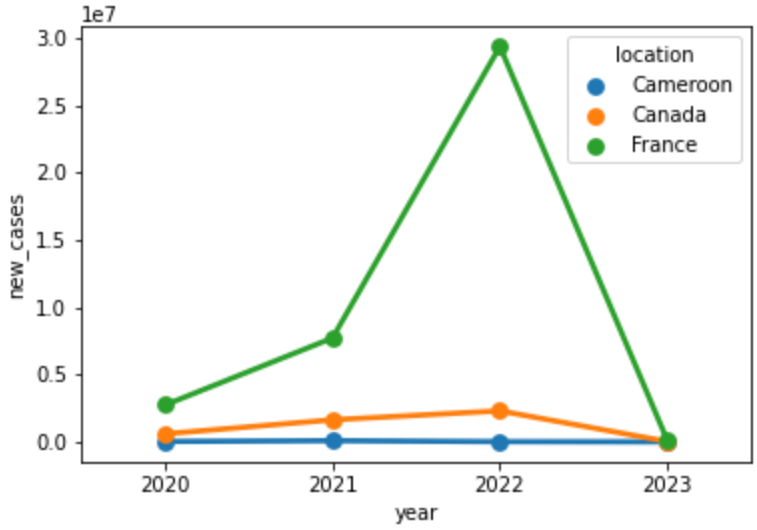


19) Total number of deaths, total number of cases by country, for each year **_2020/2021/2022/2023_**

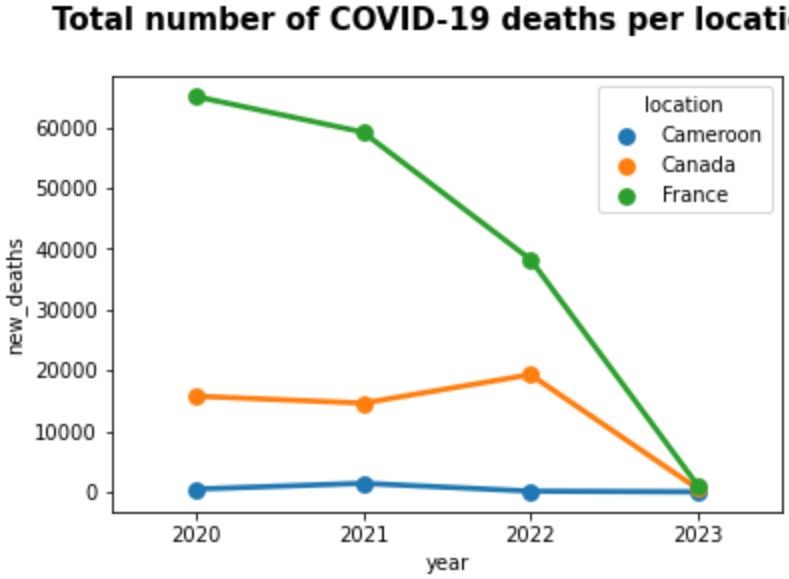
In [30]:
```
dt=data.groupby(["year","location"]).sum().loc[:,["new_cases","new_deaths"]].reset_index
```

In [31]:
```
dt_country=dt[dt["location"].isin(["France","Canada","Cameroon"])]

plt.subplot(111)
sns.pointplot(x='year', y='new_cases', data = dt_country, hue= "location")

plt.title("Total number of COVID-19 cases per location \n", fontsize=15, fontweight="bol
plt.show()
```

Total number of COVID-19 cases per location



In [32]:
```
dt_country=dt[dt["location"].isin(["France","Canada","Cameroon"])]

plt.subplot(111)
sns.pointplot(x='year', y='new_deaths', data = dt_country, hue= "location")

plt.title("Total number of COVID-19 deaths per location \n", fontsize=15, fontweight="bo
plt.show()
```



19.1) Per year, let us look at the Top 05 countires with the highest number of deaths and cases **_2020/2021/2022/2023_

In [33]:
```
dt[dt["year"]== 2020].sort_values(["new_cases","new_deaths"], na_position = "last", asce
```

Out[33]:

	year	location	new_cases	new_deaths
206	2020	United States	20217272.0	350555.0
91	2020	India	10286709.0	148995.0
27	2020	Brazil	7700828.0	195072.0
161	2020	Russia	3127347.0	56271.0
70	2020	France	2735590.0	65031.0

In [34]:
```
dt[dt["year"]== 2021].sort_values(["new_cases","new_deaths"], na_position = "last", asce
```

Out[34]:

	year	location	new_cases	new_deaths
440	2021	United States	34687377.0	475059.0
312	2021	India	24574870.0	325118.0
246	2021	Brazil	14485929.0	424262.0
439	2021	United Kingdom	10456330.0	82391.0
290	2021	France	7706191.0	59161.0

In [35]:
```
dt[dt["year"]== 2022].sort_values(["new_cases","new_deaths"], na_position = "last", asce
```

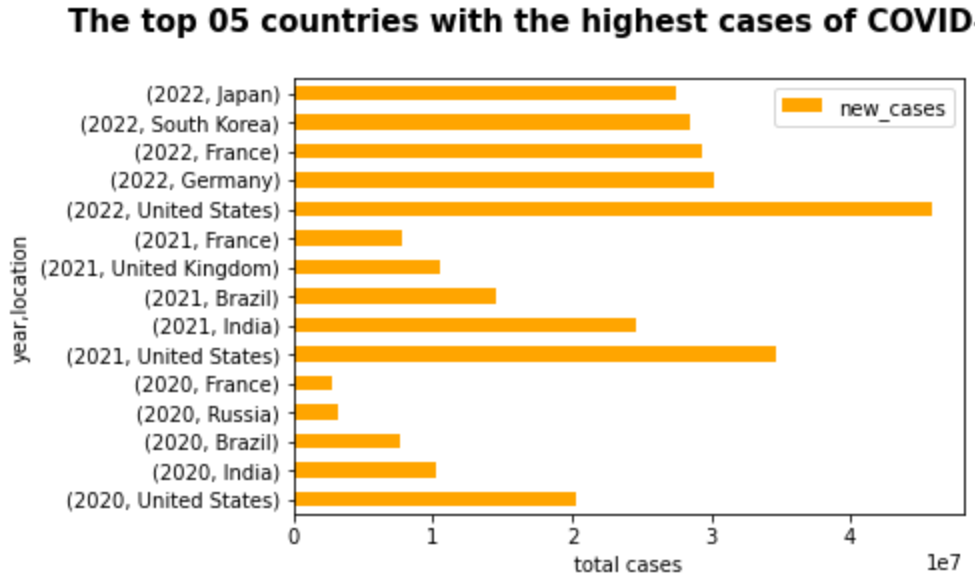
Out[35]:

	year	location	new_cases	new_deaths
673	2022	United States	45856313.0	267871.0
529	2022	Germany	30260684.0	49863.0
524	2022	France	29345799.0	38226.0
646	2022	South Korea	28481547.0	26594.0
555	2022	Japan	27501370.0	38892.0

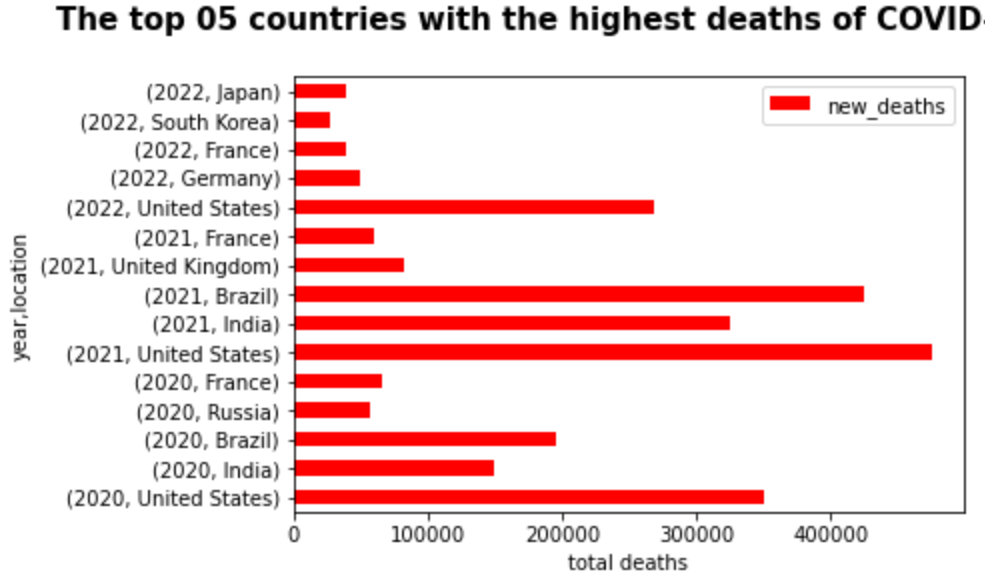
In [36]:
```
# let us concatenate the previous dataframes in order to have all the years in a bar cha

dt_2020 = dt[dt["year"]== 2020].sort_values(["new_cases","new_deaths"], na_position = "l
dt_2021 = dt[dt["year"]== 2021].sort_values(["new_cases","new_deaths"], na_position = "l
dt_2022 = dt[dt["year"]== 2022].sort_values(["new_cases","new_deaths"], na_position = "l

dt_group=dt_2020.append([dt_2021,dt_2022])
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_2404\1246131657.py:7: FutureWarning: The frame.
append method is deprecated and will be removed from pandas in a future version. Use pan
das.concat instead.
  dt_group=dt_2020.append([dt_2021,dt_2022])
```

In [37]:
```
dt_group.set_index(["year","location"]).plot(kind='barh', y=["new_cases"], color = 'oran
plt.title ("The top 05 countries with the highest cases of COVID-19 per year \n", fontsi
plt.xlabel("total cases")
plt.show()
```



In [38]:
```
dt_group.set_index(["year","location"]).plot(kind='barh', y=["new_deaths"], color = 'red
plt.title ("The top 05 countries with the highest deaths of COVID-19 per year \n",fontsi
plt.xlabel("total deaths")
plt.show()
```



In [39]:
```
dt[dt["year"]== 2023].sort_values(["new_cases","new_deaths"], na_position = "last", asce
# our data ends on the 09/01/2023 so this results does not take into consideration the 1
```

Out[39]:

	year	location	new_cases	new_deaths
787	2023	Japan	1356342.0	2892.0
898	2023	United States	524384.0	3774.0
873	2023	South Korea	482947.0	450.0
882	2023	Taiwan	225145.0	329.0
763	2023	Germany	170206.0	1510.0

20) Mortality rate

> ***Total deaths / population***

20.1) Annual mortality rate (per 1000 people)

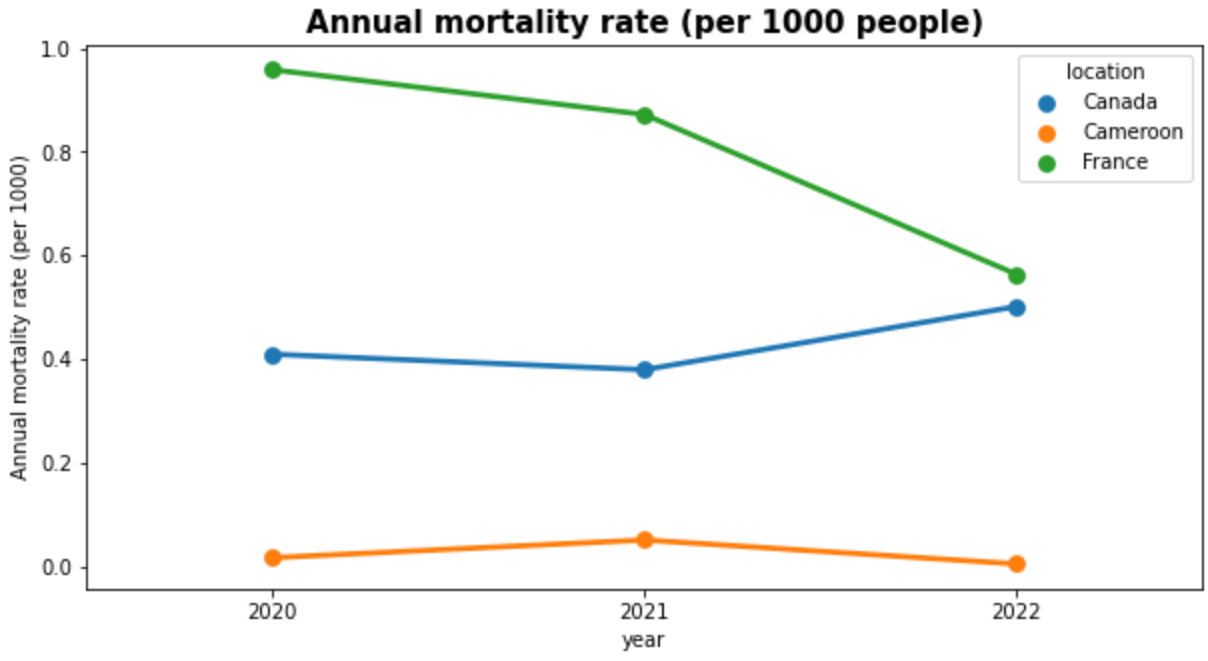
In [40]:
```
df= data.copy()
df = df.groupby(["year","iso_code","location","population"]).sum().loc[:,["new_deaths"]]
len(df.columns) #4
df.insert(5, "Annual mortality rate (per 1000)", (df["new_deaths"]/df["population"])*100
df.head(2)
```

Out[40]:

	year	iso_code	location	population	new_deaths	Annual mortality rate (per 1000)
0	2020	ABW	Aruba	106459.0	50.0	0.469664
1	2020	AFG	Afghanistan	41128772.0	2189.0	0.053223

20.2 Let us choose some countries to visualise their annual mortality rate_ (per 1000 people)

In [41]:
```
df_country= df[(df["location"].isin(["France", "Canada", "Cameroon"])) & (df["year"].isi

plt.figure(figsize=(10,5))
sns.pointplot(x="year", y="Annual mortality rate (per 1000)", data = df_country, hue = "
plt.title("Annual mortality rate (per 1000 people)",fontsize=15, fontweight="bold")
plt.show()
```



20.3 The top 05 countries with the highest annual mortality rate (per 1000 population)

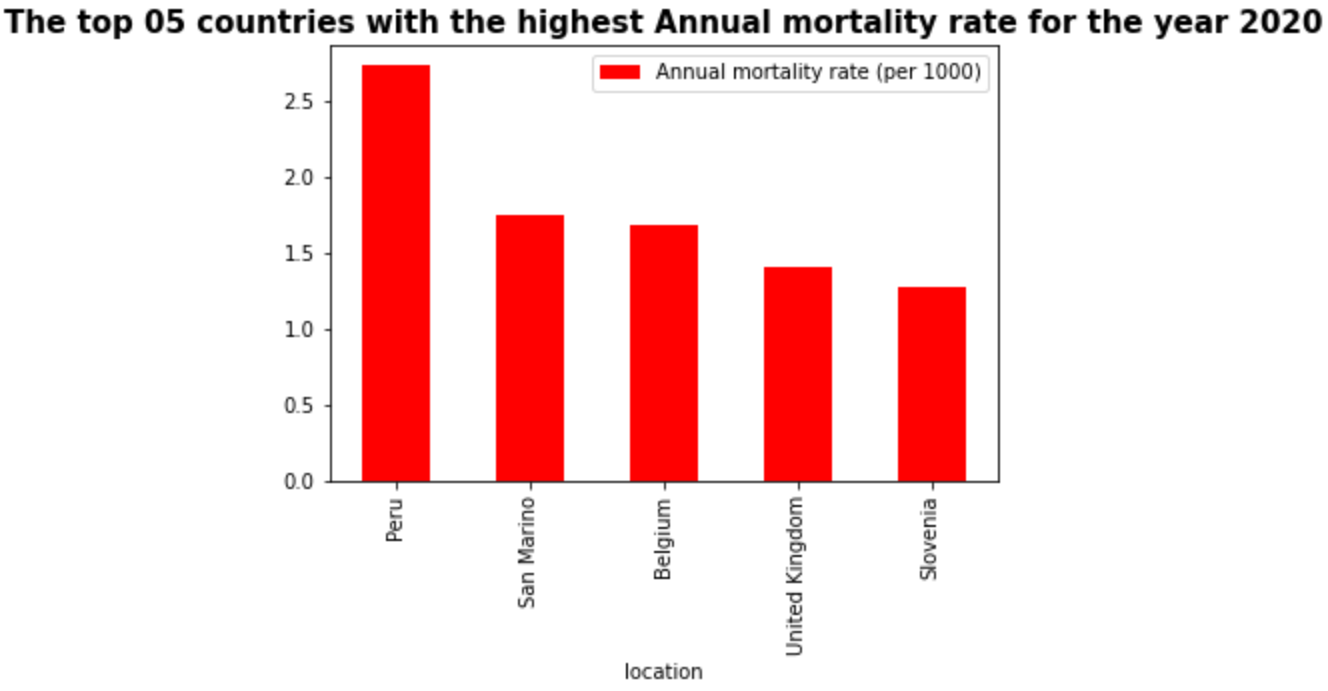
_2020/2021/2022_

In [42]:
```
df[df["year"]==2020].sort_values("Annual mortality rate (per 1000)", ascending =False, n
```

Out[42]:

	year	iso_code	location	population	new_deaths	Annual mortality rate (per 1000)
155	2020	PER	Peru	34049588.0	93070.0	2.733366
177	2020	SMR	San Marino	33690.0	59.0	1.751262
14	2020	BEL	Belgium	11655923.0	19645.0	1.685409
68	2020	GBR	United Kingdom	67508936.0	94998.0	1.407191
185	2020	SVN	Slovenia	2119843.0	2697.0	1.272264

In [43]:
```
df[df["year"]==2020].sort_values("Annual mortality rate (per 1000)", ascending =False, n
plt.title("The top 05 countries with the highest Annual mortality rate for the year 2020
plt.show()
```

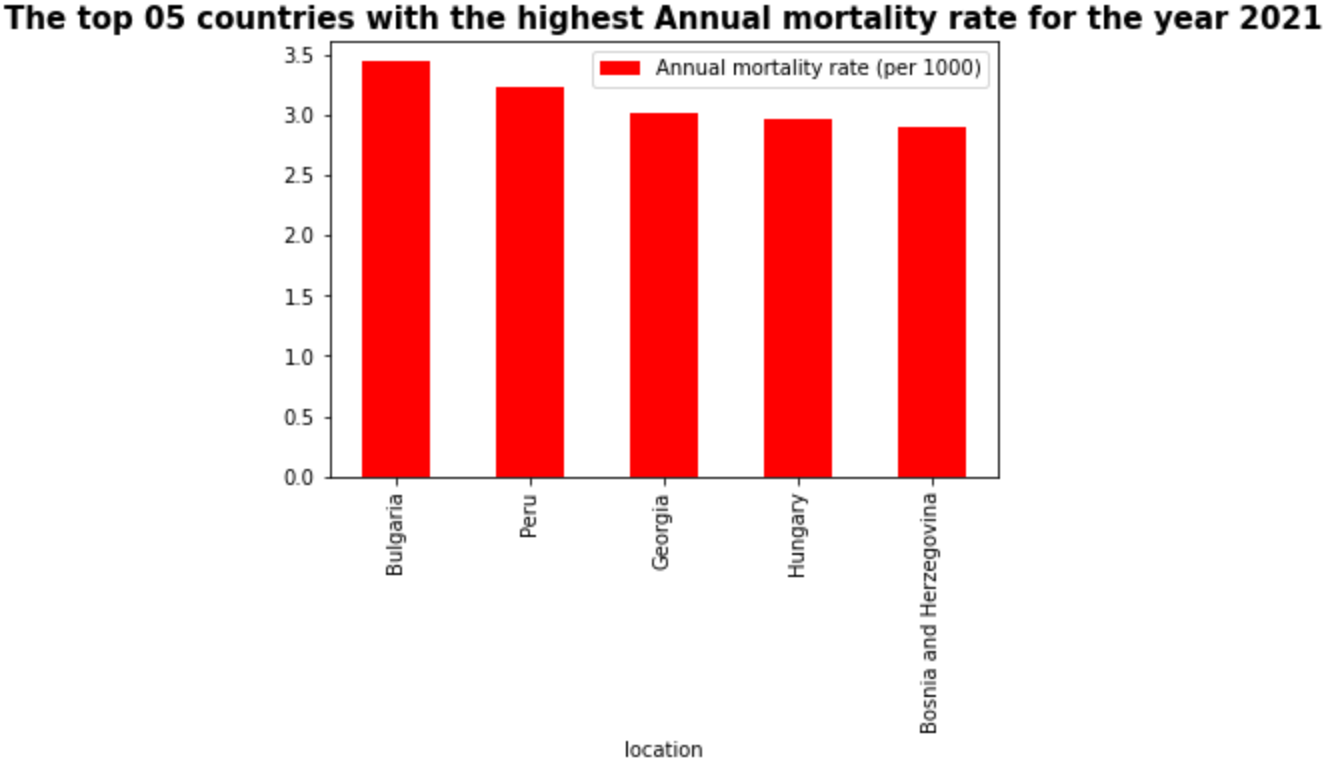


In [44]:
```
df[df["year"]==2021].sort_values("Annual mortality rate (per 1000)", ascending =False, n
```

Out[44]:

	year	iso_code	location	population	new_deaths	Annual mortality rate (per 1000)
238	2021	BGR	Bulgaria	6781955.0	23379.0	3.447236
383	2021	PER	Peru	34049588.0	110329.0	3.240245
290	2021	GEO	Georgia	3744385.0	11295.0	3.016517
308	2021	HUN	Hungary	9967304.0	29649.0	2.974626
241	2021	BIH	Bosnia and Herzegovina	3233530.0	9403.0	2.907967

In [45]:
```
df[df["year"]==2021].sort_values("Annual mortality rate (per 1000)", ascending =False, n
plt.title("The top 05 countries with the highest Annual mortality rate for the year 2021
plt.show()
```

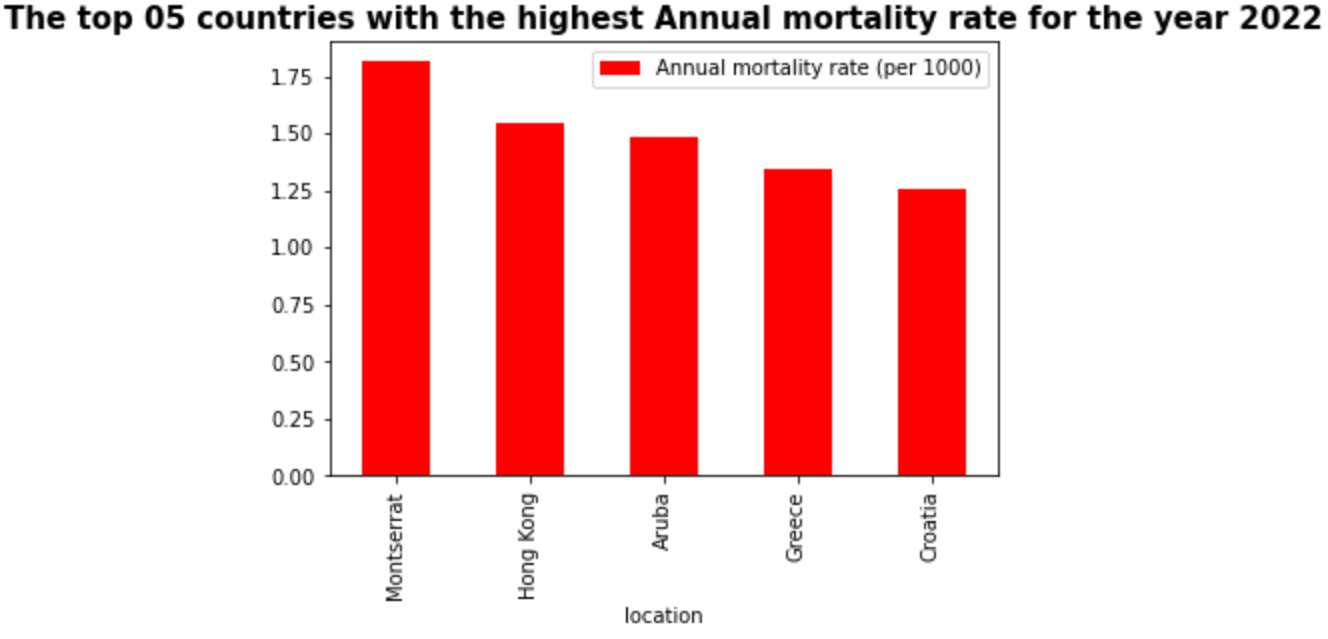


In [46]:
```
df[df["year"]==2022].sort_values("Annual mortality rate (per 1000)", ascending =False, n
```

Out[46]:

	year	iso_code	location	population	new_deaths	Annual mortality rate (per 1000)
592	2022	MSR	Montserrat	4413.0	8.0	1.812826
539	2022	HKG	Hong Kong	7488863.0	11594.0	1.548166
453	2022	ABW	Aruba	106459.0	158.0	1.484139
533	2022	GRC	Greece	10384972.0	13989.0	1.347043
541	2022	HRV	Croatia	4030361.0	5058.0	1.254974

In [47]:
```
df[df["year"]==2022].sort_values("Annual mortality rate (per 1000)", ascending =False, n
plt.title("The top 05 countries with the highest Annual mortality rate for the year 2022
plt.show()
```



21) Infection rate

> **_total*cases / population***

21.1) Monthly infection rate

In [48]:
```
df= data.copy()
df.head(2)
```

Out[48]:

	iso_code	continent	location	month	year	date	total_cases	new_cases	total_deaths	new_deaths	total_
0	AFG	Asia	Afghanistan	2	2020	2020-02-24	5.0	5.0	NaN	NaN	
1	AFG	Asia	Afghanistan	2	2020	2020-02-25	5.0	0.0	NaN	NaN	

In [49]:
```
df = df.groupby(["year","month","location","population"]).sum().loc[:,["new_cases"]].res
df.insert(5,"Monthly infection rate",(df["new_cases"]/ df["population"])*100)
```

In [50]:
```
df.head(3)
```

Out[50]:

	year	month	location	population	new_cases	Monthly infection rate
0	2020	1	Argentina	45510324.0	0.0	0.000000
1	2020	1	Australia	26177410.0	9.0	0.000034
2	2020	1	Cambodia	16767851.0	1.0	0.000006

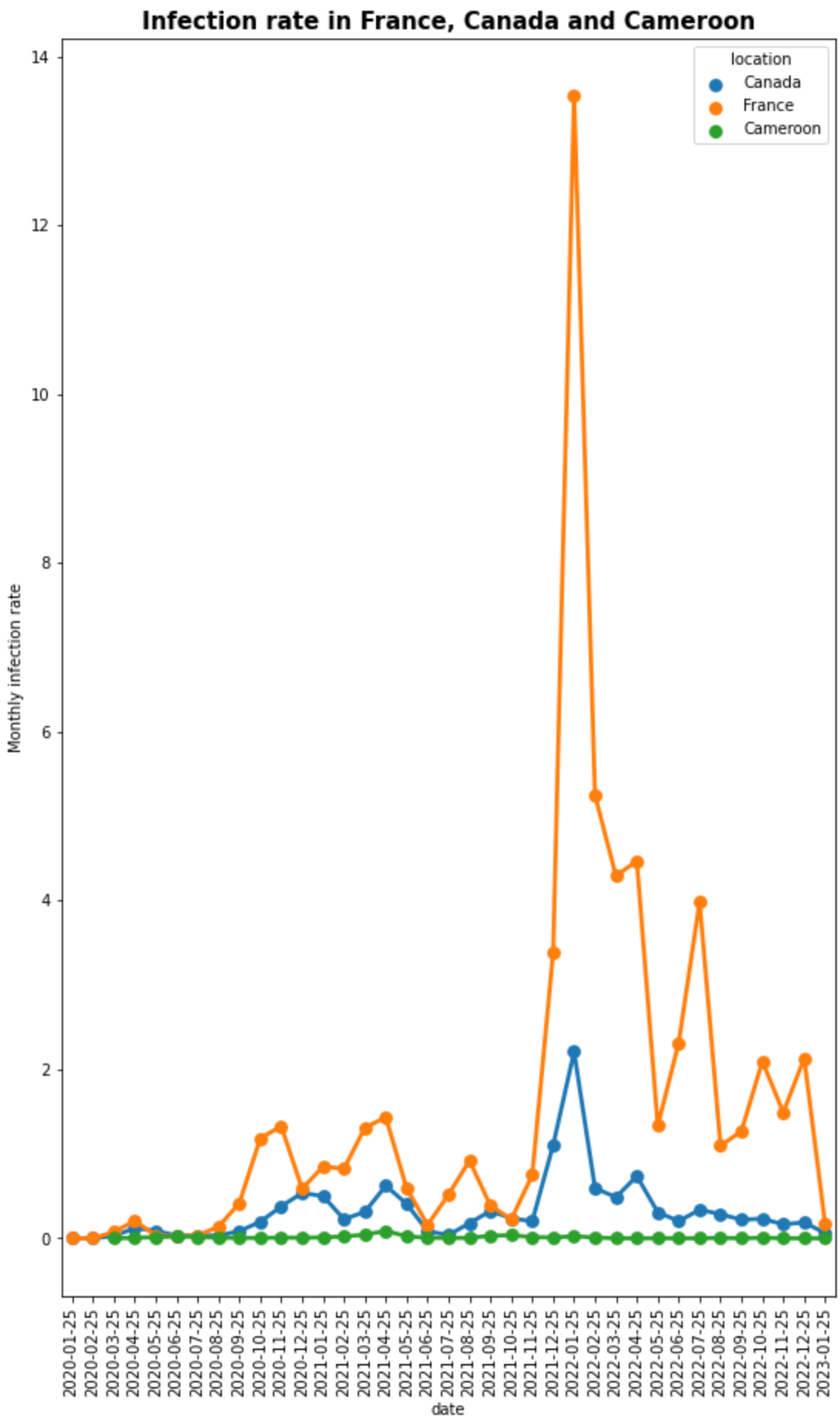
21.2) Monthly infection rate in France, Canada, Cameroon

> Let us create a date column to visualise the monthly infection throughout the years

In [51]:
```
df["day"]= 25  # we just create a day column inorder to combine it later with month and
df["date"] = (df["year"].astype(str) + "/" + df["month"].astype(str) + "/" + df["day"].a
```

In [52]:
```
df["date"]= pd.to_datetime(df["date"]).dt.date
```

In [53]:
```
df_country = df[df["location"].isin(["France","Canada","Cameroon"])]
pd.to_datetime(df["date"]).dt.date

plt.figure(figsize=(9,15))
sns.pointplot(x="date", y="Monthly infection rate", data = df_country , hue = "location"

plt.title("Infection rate in France, Canada and Cameroon",fontsize=15, fontweight="bold"
plt.xticks(rotation=90)
plt.show()
```



21.3 Top 05 countries with the highest infection rate for the years **_2020/2021/2022_**

In [54]:
```
df[df["year"]==2020].nlargest(5, columns = ["Monthly infection rate"], keep = "all")
```

Out[54]:

	year	month	location	population	new_cases	Monthly infection rate	day	date

1590	2020	10	Andorra	79843.0	2706.0	3.389151	25	2020-10-25
2100	2020	12	Gibraltar	32677.0	1020.0	3.121462	25	2020-12-25
2137	2020	12	Lithuania	2750058.0	82554.0	3.001900	25	2020-12-25
2201	2020	12	Slovakia	5643455.0	153256.0	2.715641	25	2020-12-25
1919	2020	11	Luxembourg	647601.0	17544.0	2.709075	25	2020-11-25

In [55]:
```
df[df["year"]==2021].nlargest(5, columns = ["Monthly infection rate"], keep = "all")
```

Out[55]:

	year	month	location	population	new_cases	Monthly infection rate	day	date
4773	2021	12	Andorra	79843.0	6625.0	8.297534	25	2021-12-25
4573	2021	11	Cayman Islands	68722.0	5424.0	7.892669	25	2021-11-25
3629	2021	7	British Virgin Islands	31332.0	2202.0	7.027959	25	2021-07-25
3262	2021	5	Maldives	523798.0	34561.0	6.598154	25	2021-05-25
3906	2021	8	French Polynesia	306292.0	20130.0	6.572160	25	2021-08-25

In [56]:
```
df[df["year"]==2022].nlargest(5, columns = ["Monthly infection rate"], keep = "all")
```

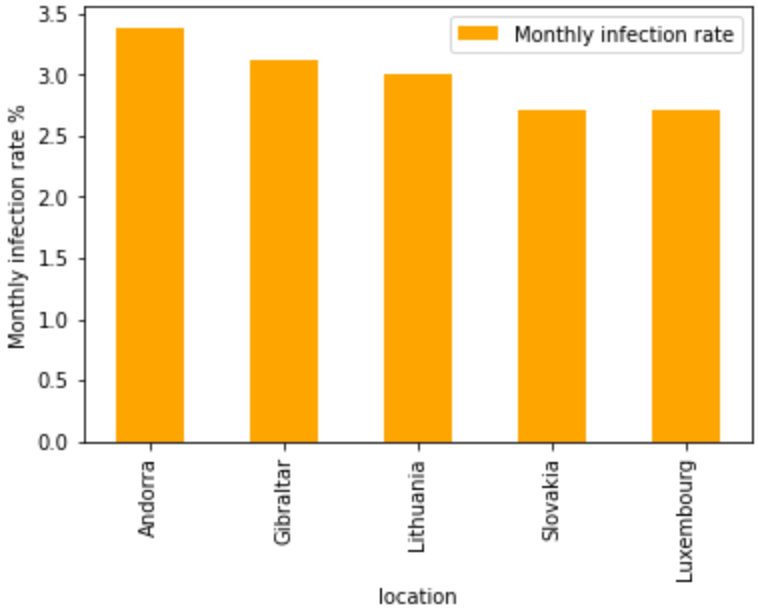
Out[56]:

	year	month	location	population	new_cases	Monthly infection rate	day	date
6004	2022	5	Falkland Islands	3801.0	1510.0	39.726388	25	2022-05-25
6758	2022	8	Marshall Islands	41593.0	14978.0	36.010867	25	2022-08-25
7031	2022	9	Saint Helena	5401.0	1514.0	28.031846	25	2022-09-25
5304	2022	2	Faeroe Islands	53117.0	14612.0	27.509084	25	2022-02-25
6310	2022	6	Nauru	12691.0	3391.0	26.719723	25	2022-06-25

In [57]:
```
df_2020 = df[df["year"]==2020].nlargest(5, columns = ["Monthly infection rate"], keep =
df_2021= df[df["year"]==2021].nlargest(5, columns = ["Monthly infection rate"], keep = "
df_2022= df[df["year"]==2022].nlargest(5, columns = ["Monthly infection rate"], keep = "

plt.figure(figsize=(6,6))
df_2020.set_index("location").plot(kind = 'bar', y=["Monthly infection rate"], color = "
plt.ylabel("Monthly infection rate %")
plt.title(" Top 05 countries with the highest infection rate for the year 2020 \n",fonts

plt.figure(figsize=(6,6))
df_2021.set_index("location").plot(kind = 'bar', y=["Monthly infection rate"], color = "
plt.ylabel("Monthly infection rate %")
plt.title(" Top 05 countries with the highest infection rate for the year 2021 \n",fonts


plt.figure(figsize=(6,6))
df_2022.set_index("location").plot(kind = 'bar', y=["Monthly infection rate"], color = "
plt.ylabel("Monthly infection rate %")
plt.title(" Top 05 countries with the highest infection rate for the year 2022 \n",fonts

plt.show()
```

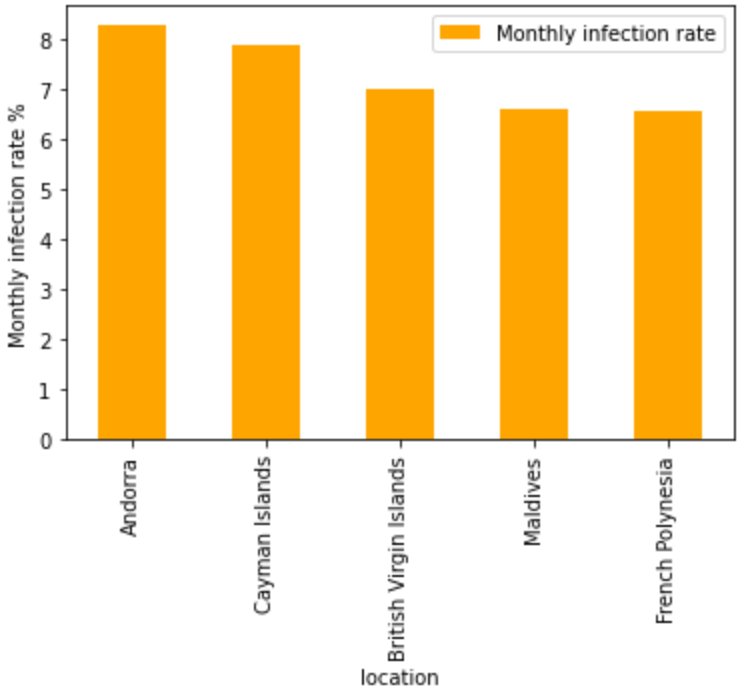
```
<Figure size 432x432 with 0 Axes>
```

Top 05 countries with the highest infection rate for the year 2020



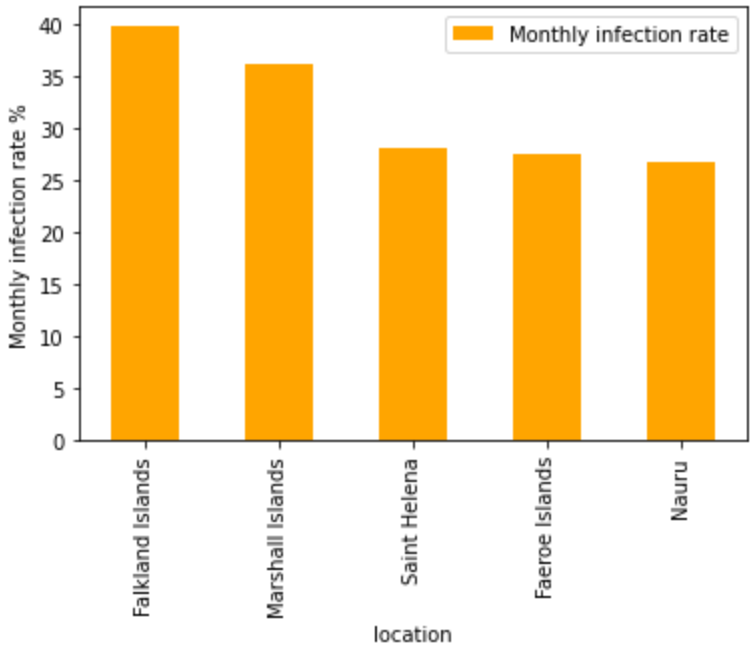

```
<Figure size 432x432 with 0 Axes>
```

Top 05 countries with the highest infection rate for the year 2021




```
<Figure size 432x432 with 0 Axes>
```

Top 05 countries with the highest infection rate for the year 2022



22) Average monthly infection and mortality

> I will do the **_sum of new*cases*** (infected) and **_new*deaths*** per ***month*** for a given ***year***, ***month*** and ***location*** using ***pivot tables***
> I will later on do a ***groupby()*** by year and location to ***add up all the monthly cases and deaths*** per ***location*** Lastly i divide the **_total*cases*** and **_total*deaths*** by ***12 months*** to have the ***average monthly cases and deaths of COVID-19***

In [58]:
```
df = pd.pivot_table(data, values=['new_cases', 'new_deaths','date'], index=['year','mont
                    aggfunc={'new_cases': np.sum,
                             'new_deaths': np.sum,
                              'date': 'count'})
df = df.reset_index()
```

In [59]:
```
df.head(3)
```

Out[59]:

	year	month	location	iso_code	date	new_cases	new_deaths
0	2020	1	Argentina	ARG	31	0.0	0.0
1	2020	1	Australia	AUS	6	9.0	0.0
2	2020	1	Cambodia	KHM	5	1.0	0.0

In [60]:
```
df = df.groupby(["year","location",'iso_code']).sum().loc[:,["new_cases","new_deaths"]].
```

In [61]:
```
df.insert(5, "Monthly average cases", df["new_cases"]/12)
df.insert(6, "Monthly average deaths", df["new_deaths"]/12)
```

22.1) Let us remove the records that contain the year 2023

> The year 2023 just has records for January, not for all the months of the year.It isn't interesting to keep it

In [62]:
```
df = df[~(df["year"]==2023)]
```

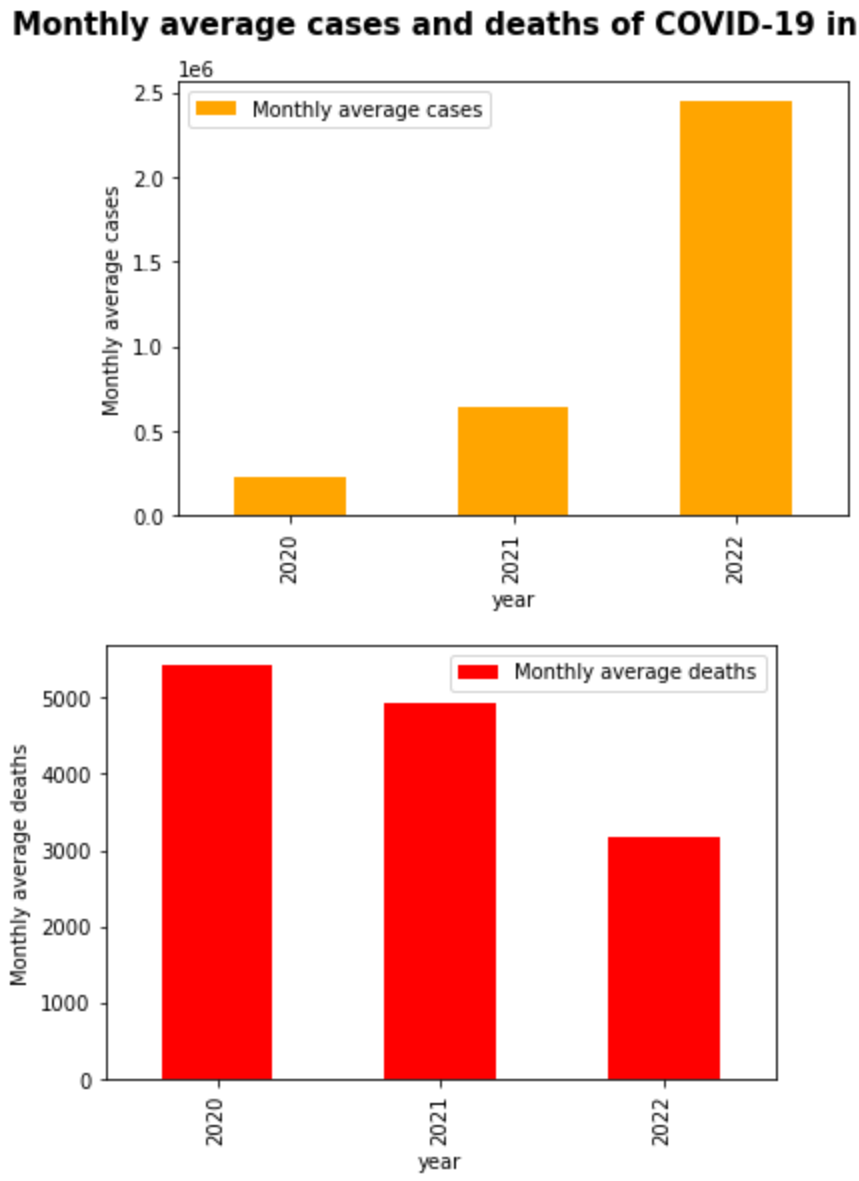
22.2 Let us look at France, Canada and Cameroon

In [63]:
```
df[df["location"]== "France"]
```

Out[63]:

	year	location	iso_code	new_cases	new_deaths	Monthly average cases	Monthly average deaths
70	2020	France	FRA	2735590.0	65031.0	2.279658e+05	5419.250000
290	2021	France	FRA	7706191.0	59161.0	6.421826e+05	4930.083333
524	2022	France	FRA	29345799.0	38226.0	2.445483e+06	3185.500000

In [64]:
```
df[df["location"]== "France"].set_index("year").plot(kind = "bar", y =["Monthly average
plt.ylabel("Monthly average cases")

plt.title("Monthly average cases and deaths of COVID-19 in FRANCE \n",fontsize=15, fontw

df[df["location"]== "France"].set_index("year").plot(kind = "bar", y =["Monthly average
plt.ylabel("Monthly average deaths")

plt.show()
```



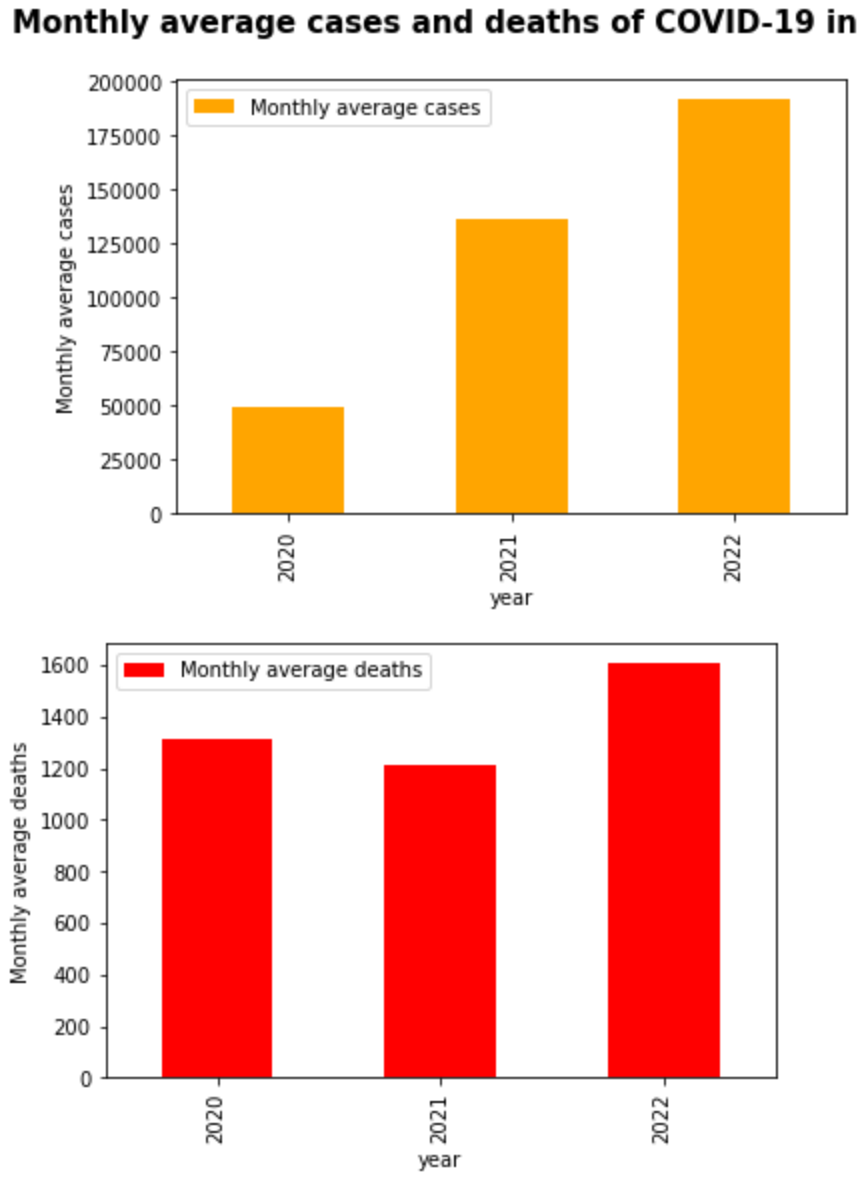
In [65]:
```
df[df["location"]== "Canada"]
```

Out[65]:

	year	location	iso_code	new_cases	new_deaths	Monthly average cases	Monthly average deaths
35	2020	Canada	CAN	590249.0	15737.0	49187.416667	1311.416667
254	2021	Canada	CAN	1633486.0	14585.0	136123.833333	1215.416667
488	2022	Canada	CAN	2297368.0	19276.0	191447.333333	1606.333333

In [66]:

```
df[df["location"]== "Canada"].set_index("year").plot(kind = "bar", y =["Monthly average
plt.ylabel("Monthly average cases")

plt.title("Monthly average cases and deaths of COVID-19 in CANADA\n",fontsize=15, fontwe

df[df["location"]== "Canada"].set_index("year").plot(kind = "bar", y =["Monthly average
plt.ylabel("Monthly average deaths")

plt.show()
```



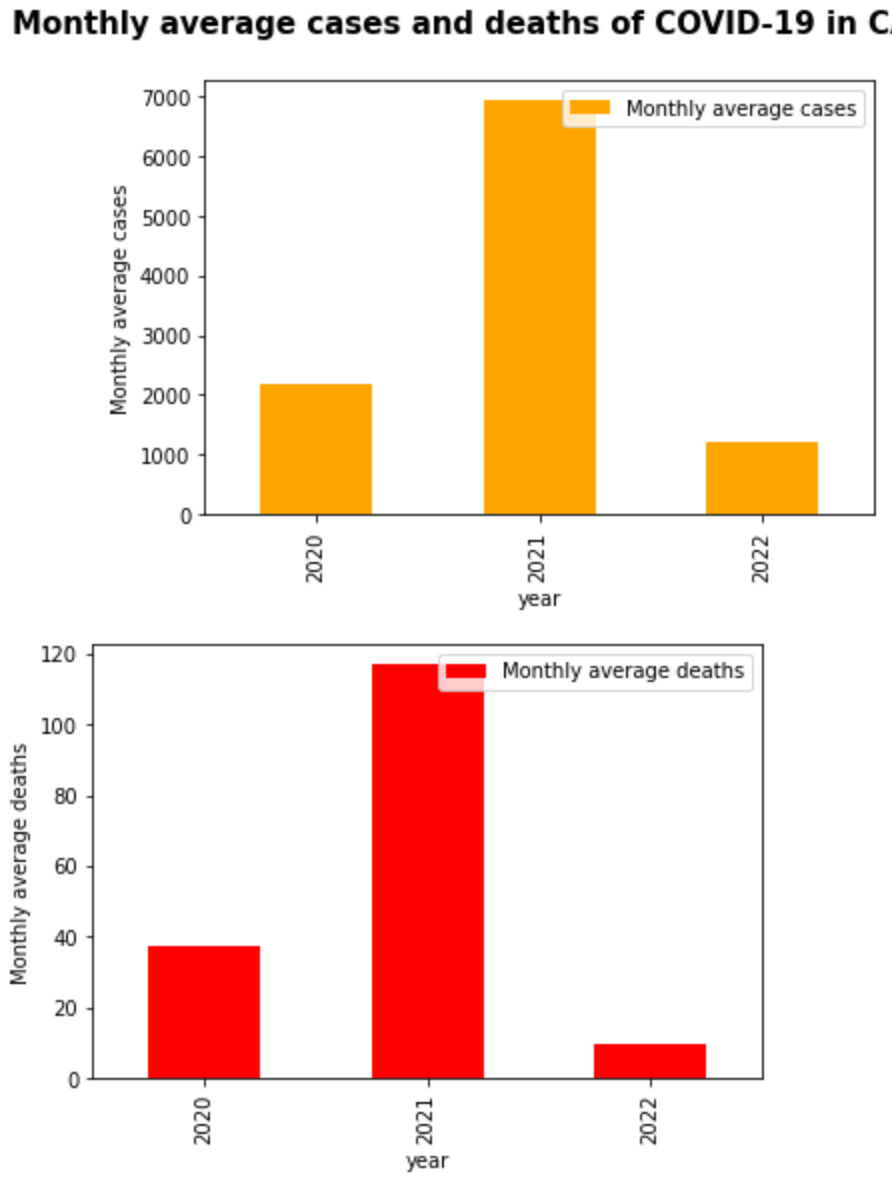
In [67]:

```
df[df["location"]== "Cameroon"]
```

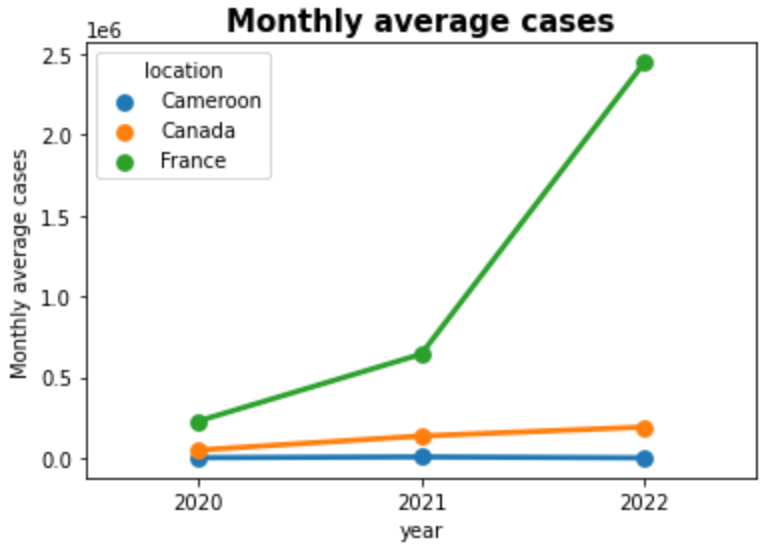
Out[67]:

	year	location	iso_code	new_cases	new_deaths	Monthly average cases	Monthly average deaths
34	2020	Cameroon	CMR	26277.0	448.0	2189.750000	37.333333
253	2021	Cameroon	CMR	83090.0	1403.0	6924.166667	116.916667
487	2022	Cameroon	CMR	14626.0	114.0	1218.833333	9.500000

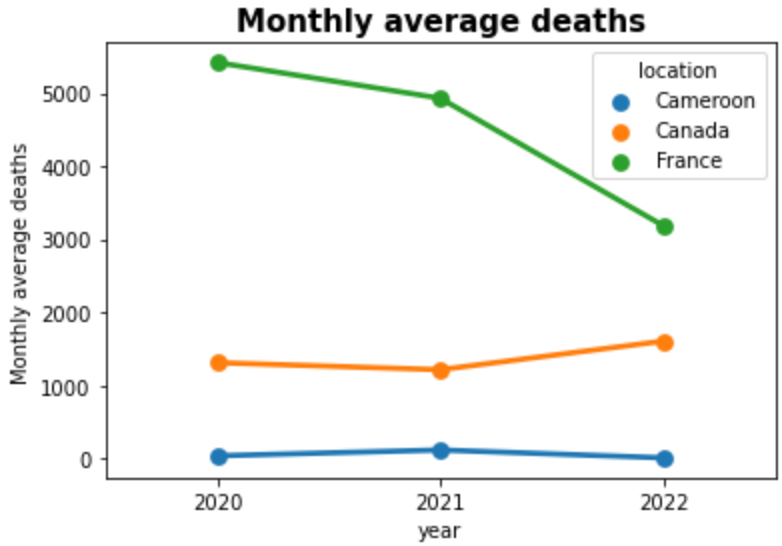
In [68]:
```
df[df["location"]== "Cameroon"].set_index("year").plot(kind = "bar", y =["Monthly averag
plt.ylabel("Monthly average cases")

plt.title("Monthly average cases and deaths of COVID-19 in CAMAROON\n",fontsize=15, font

df[df["location"]== "Cameroon"].set_index("year").plot(kind = "bar", y =["Monthly averag
plt.ylabel("Monthly average deaths")

plt.show()
```



In [69]:
```
df_loc=df[df["location"].isin(["Cameroon","France","Canada"])]

sns.pointplot(x="year",y = "Monthly average cases", data = df_loc, hue = "location")
plt.title("Monthly average cases",fontsize=15, fontweight="bold")
plt.show()
```



In [70]:
```
sns.pointplot(x="year",y = "Monthly average deaths", data = df_loc, hue = "location")
plt.title("Monthly average deaths",fontsize=15, fontweight="bold")
plt.show()
```



23) Vaccinations

> (people vaccinated / population)* 100%.
> Let us keep in mind that the columns _"**people*vaccinated***" and "***people fully vaccinated***" are ***cumulative columns***.

In [71]:
```
data.head(2)
```

Out[71]:

	iso_code	continent	location	month	year	date	total_cases	new_cases	total_deaths	new_deaths	total_
0	AFG	Asia	Afghanistan	2	2020	2020-02-24	5.0	5.0	NaN	NaN	
1	AFG	Asia	Afghanistan	2	2020	2020-02-25	5.0	0.0	NaN	NaN	

> When going through the data we realise that we have missing values for some months of some locations

> The columns we are interested in are cumulative columns,for each location we will extract the max number of people vaccinated and people fully vaccinated before applying an operation

****23.1) For each location let us look at the cumulative trend in vaccinations throughout the years****

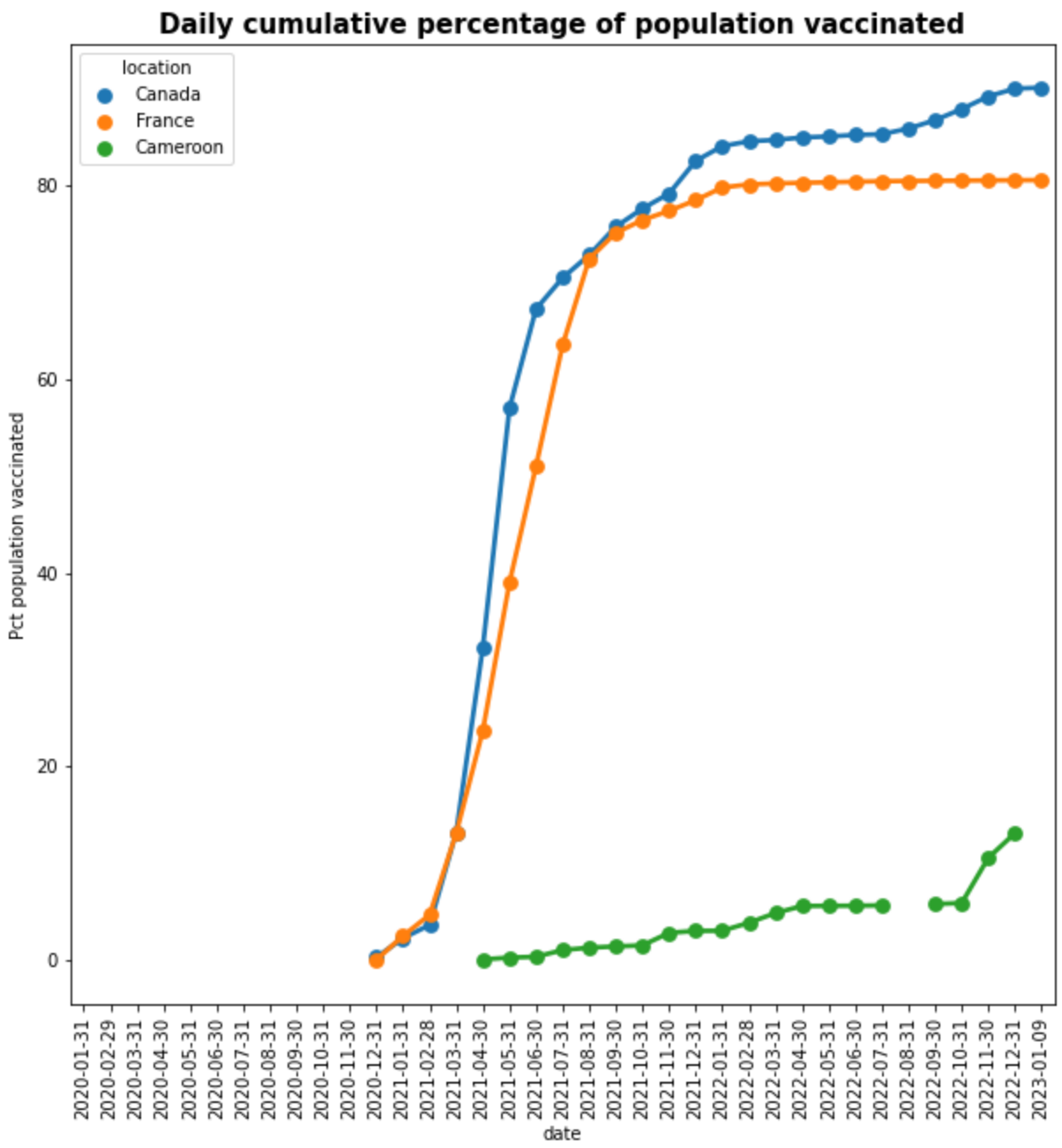
In [72]:
```
df=data.groupby(["year","month","location",'iso_code',"population"]).max().loc[:,["peopl
```

In [73]:
```
df.insert(7,"Pct population vaccinated", (df["people_vaccinated"]/df["population"])*100)
df.insert(8,"Pct population fully vaccinated", (df["people_fully_vaccinated"]/df["popula
```

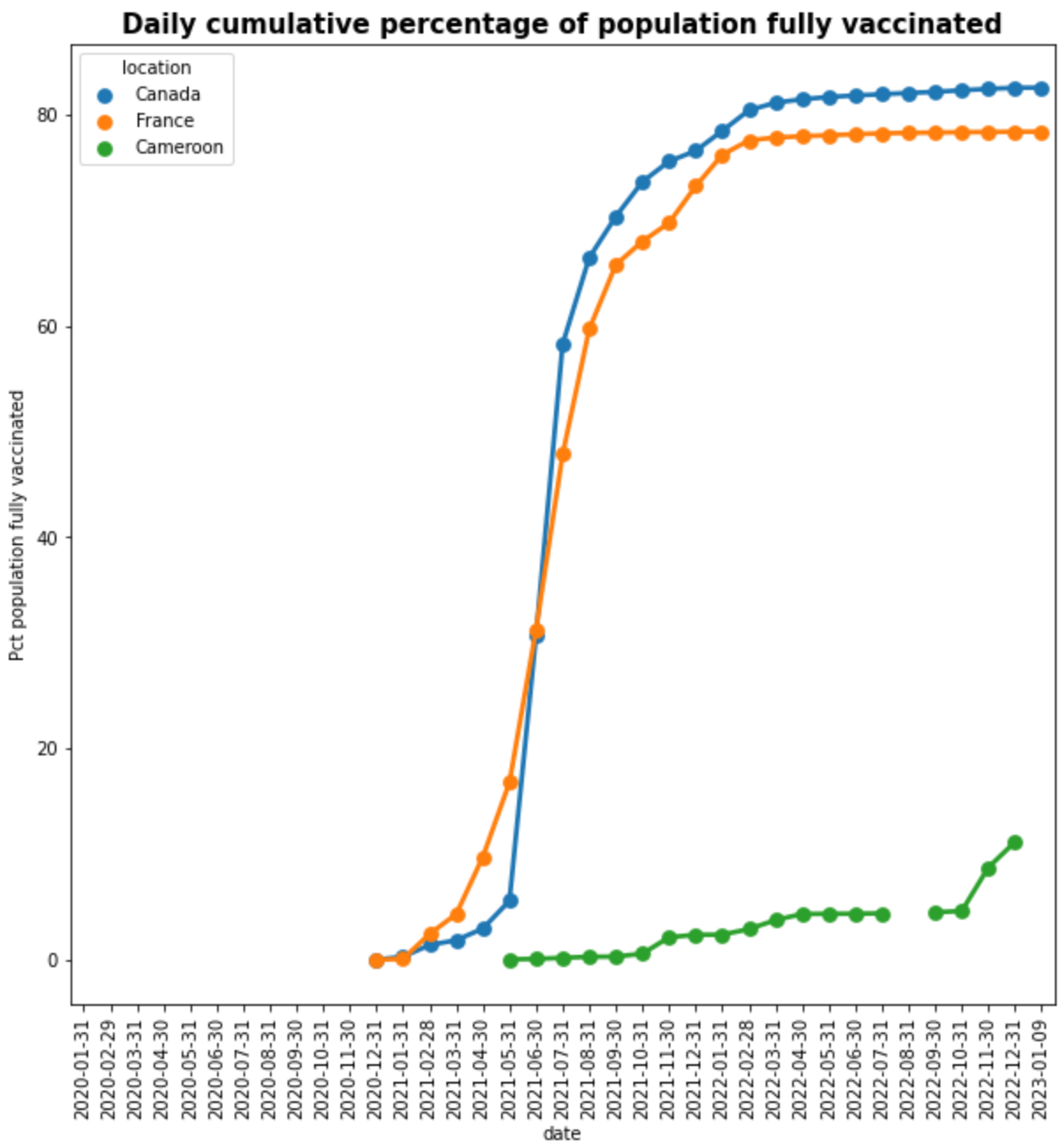
23.2) ***Convert the datetime column to date***

In [74]:
```
df['date'] = pd.to_datetime(df['date']).dt.date
```

In [75]:
```
df_loc= df[df["location"].isin(["France","Canada","Cameroon"])]

plt.figure(figsize=(10,10))
sns.pointplot(x="date",y = "Pct population vaccinated", data = df_loc, hue = "location")
plt.title("Daily cumulative percentage of population vaccinated",fontsize=15, fontweight
plt.xticks(rotation = 90)
plt.show()
```



In [76]:
```
df_loc= df[df["location"].isin(["France","Canada","Cameroon"])]

plt.figure(figsize=(10,10))
sns.pointplot(x="date",y = "Pct population fully vaccinated", data = df_loc, hue = "loca
plt.title("Daily cumulative percentage of population fully vaccinated",fontsize=15, font
plt.xticks(rotation = 90)
plt.show()
```



23.3) **Choropleth map**

> Let us keep in mind that the columns _"**people*vaccinated***" and "***people fully vaccinated***" are ***cumulative columns***

In [87]:
```
df=data.groupby(["year","location",'iso_code',"population"]).max().loc[:,["people_vaccin
```

In [88]:
```
df[df["year"]==2022].sort_values( "people_vaccinated", axis = 0,na_position= "first", as
# some countries do not have records for some years
```

Out[88]:

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated
477	2022	Bonaire Sint Eustatius and Saba	BES	27052.0	NaN	NaN
516	2022	Eritrea	ERI	3684041.0	NaN	NaN

In [89]:
```
# people vaccinated should be less than or equal to the population, let us verify
df[df["people_vaccinated"]>df["population"]]
# !!!! we will delete these records
```

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated

Out[89]:

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated
297	2021	Gibraltar	GIB	32677.0	41173.0	40065.0
438	2021	United Arab Emirates	ARE	9441138.0	9881456.0	9059559.0
482	2022	Brunei	BRN	449002.0	450404.0	445929.0
531	2022	Gibraltar	GIB	32677.0	42175.0	41465.0
621	2022	Qatar	QAT	2695131.0	2850159.0	2850158.0
661	2022	Tokelau	TKL	1893.0	2203.0	2203.0
671	2022	United Arab Emirates	ARE	9441138.0	9991089.0	9792266.0

In [90]:
```
df[(df["people_fully_vaccinated"]>df["population"])]
# !!! we will delete these records
```

Out[90]:

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated
297	2021	Gibraltar	GIB	32677.0	41173.0	40065.0
531	2022	Gibraltar	GIB	32677.0	42175.0	41465.0
621	2022	Qatar	QAT	2695131.0	2850159.0	2850158.0
661	2022	Tokelau	TKL	1893.0	2203.0	2203.0
671	2022	United Arab Emirates	ARE	9441138.0	9991089.0	9792266.0

In [91]:
```
df= df[~((df["people_fully_vaccinated"]>df["population"]) | (df["people_vaccinated"]>df[
```

In [92]:
```
df.insert(6,"Pct population vaccinated", (df["people_vaccinated"]/df["population"])*100)
df.insert(7,"Pct population fully vaccinated", (df["people_fully_vaccinated"]/df["popula
```

In [93]:
```
df[df["Pct population vaccinated"]>100]  # we have no records we are good
```

Out[93]:

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated	Pct population vaccinated	Pct population fully vaccinated

In [94]:
```
df[df["Pct population fully vaccinated"]>100]  # we have no records we are good
```

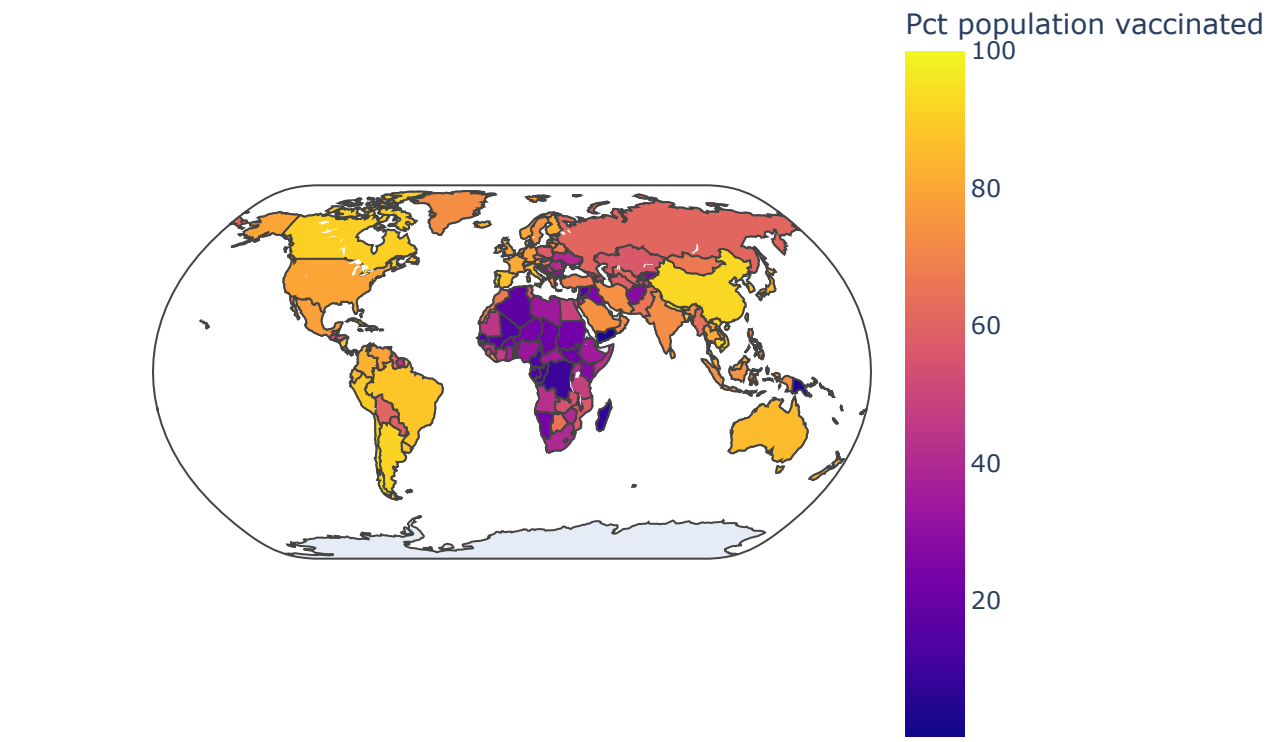
Out[94]:

	year	location	iso_code	population	people_vaccinated	people_fully_vaccinated	Pct population vaccinated	Pct population fully vaccinated

In [95]:
```
# for each location let us select the highest Pct population vaccinated... since we are
df = df.groupby(["iso_code","location"]).max().loc[:,["Pct population vaccinated","Pct p
```

In [98]:
```
import plotly.express as px
import plotly.offline as pyo
pyo.init_notebook_mode()
fig = px.choropleth(df,
                    locations="iso_code", # colunm containing ISO 3166 country codes
                    color="Pct population vaccinated", # column by which to color-code
                    hover_name="location", # column to display in hover information
                   color_continuous_scale=px.colors.sequential.Plasma)

fig.update_layout(
```

```
    # add a title text for the plot
    title_text = 'Percentage of population vaccinated up to the 09/01/2023',
    #geo_scope = 'africa', # can be set to north america | south america | africa | asia
    geo = dict(projection={'type':'natural earth'}) # by default, projection type is set
)
fig.show()
```



Attached online document

> Online resource for geospatial map