Reto analítica

Con el siguiente fragmento de código, hacemos la lectura del archivo con pandas y lo visualizamos. Tambien importamos la libreria para graficar.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
import sklearn
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import scale
import sklearn.metrics as sm
from sklearn import datasets
from sklearn.metrics import confusion_matrix, classification_report
df = pd.read_csv ('country_vaccinations.csv')
df
```

Out[2]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinati
0	Albania	ALB	2021-01-10	0.0	0.0	NaN	
1	Albania	ALB	2021-01-11	NaN	NaN	NaN	
2	Albania	ALB	2021-01-12	128.0	128.0	NaN	
3	Albania	ALB	2021-01-13	188.0	188.0	NaN	
4	Albania	ALB	2021-01-14	266.0	266.0	NaN	
...	...	...	...	...	...	...	
6740	Zimbabwe	ZWE	2021-03-12	36283.0	36283.0	NaN	
6741	Zimbabwe	ZWE	2021-03-13	36359.0	36359.0	NaN	
6742	Zimbabwe	ZWE	2021-03-14	36359.0	36359.0	NaN	
6743	Zimbabwe	ZWE	2021-03-15	37660.0	37660.0	NaN	
6744	Zimbabwe	ZWE	2021-03-16	39550.0	39550.0	NaN	

6745 rows × 15 columns

A continuacion encontraremos el número mínimo y máximo de vacunaciones por país

In [3]:
```
res = df.groupby("country")["total_vaccinations"].agg(minimo="min", maximo="max")
res
```

Out[3]:

country	minimo	maximo
Albania	0.0	23635.0
Algeria	0.0	75000.0
Andorra	576.0	4914.0
Angola	0.0	6169.0
Anguilla	0.0	5000.0
...	...	...
Uruguay	366.0	215780.0
Venezuela	0.0	12194.0
Vietnam	0.0	20695.0
Wales	8257.0	1412849.0
Zimbabwe	0.0	39550.0

137 rows × 2 columns

Encontramos la media del total de vacunaciones en todos los paises

In [4]:
```
media = df['total_vaccinations'].mean()
media
```

Out[4]: 2285175.2670280463

Encontramos el promedio de vacunaciones en todos los paises

In [5]:
```
prom = df['total_vaccinations'].median()
prom
```

Out[5]: 250000.0

Encontramos la desviación estandar

In [6]:
```
std = df['total_vaccinations'].std(ddof=0)
std
```

Out[6]: 8040984.8201448675

Al analizar estos datos, podemos notar que aunque la cantidad de vacunaciones en promedio que existen es relativamente grande, al contrastarla con los mínimos y máximos, podemos ver que aun hay muchos paises que se encuentran alejados de este promedio. Analizando el contraste con la desviación estandar, podemos ver que esta diferencía es muy extensa.

Importamos la libreria de graficación

In [7]:
```
df.describe()
```

Out[7]:

	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations
count	4.243000e+03	3.785000e+03	2.576000e+03	3.576000e+03	6.550000e+03
mean	2.285175e+06	1.858055e+06	7.499273e+05	8.538608e+04	5.923553e+04
std	8.041933e+06	5.975596e+06	3.021858e+06	2.596380e+05	1.967179e+05
min	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00
25%	3.665000e+04	3.477200e+04	1.738800e+04	2.625750e+03	1.089000e+03
50%	2.500000e+05	2.303290e+05	9.256600e+04	1.344650e+04	6.424000e+03
75%	1.175071e+06	9.420170e+05	4.489362e+05	5.752425e+04	2.823125e+04
max	1.107379e+08	7.213562e+07	3.904234e+07	4.575496e+06	2.541597e+06

In [8]:
```
type(df)
```

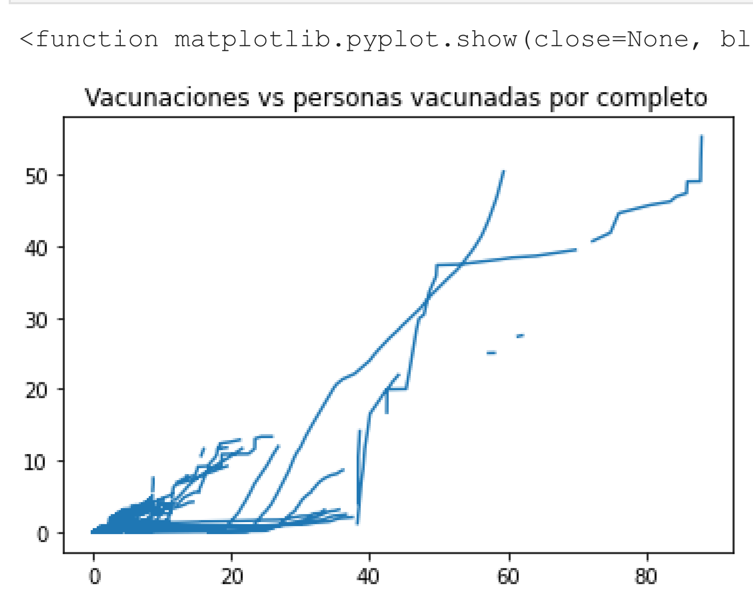
Out[8]: pandas.core.frame.DataFrame

In [9]:
```
df.total_vaccinations
```

Out[9]:
```
0           0.0
1           NaN
2         128.0
3         188.0
4         266.0
         ...
6740    36283.0
6741    36359.0
6742    36359.0
6743    37660.0
6744    39550.0
Name: total_vaccinations, Length: 6745, dtype: float64
```

In [10]:
```
plt.plot(df.people_vaccinated_per_hundred,df.people_fully_vaccinated_per_hundred)
plt.title('Vacunaciones vs personas vacunadas por completo')
plt.show
```

Out[10]: <function matplotlib.pyplot.show(close=None, block=None)>



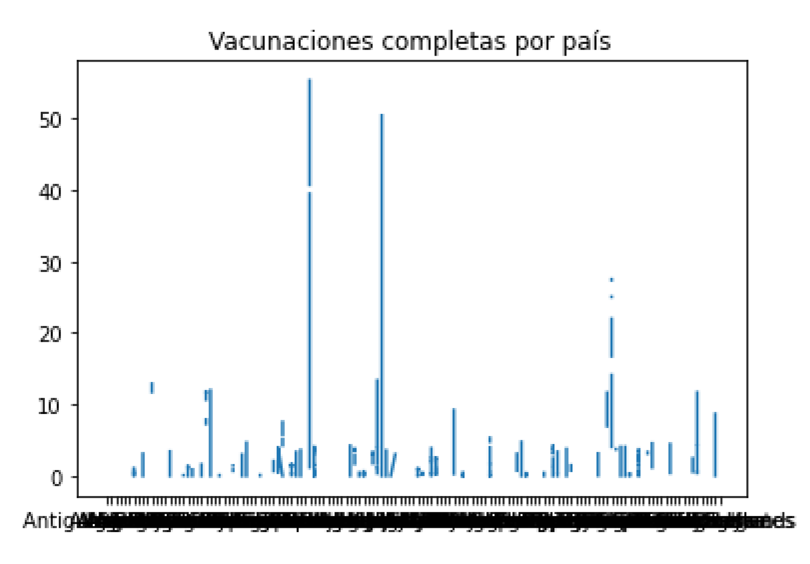
In [11]:
```
df.head()
```

Out[11]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_ra
0	Albania	ALB	2021-01-10	0.0	0.0	NaN	Na
1	Albania	ALB	2021-01-11	NaN	NaN	NaN	Na
2	Albania	ALB	2021-01-12	128.0	128.0	NaN	Na
3	Albania	ALB	2021-01-13	188.0	188.0	NaN	60
4	Albania	ALB	2021-01-14	266.0	266.0	NaN	78

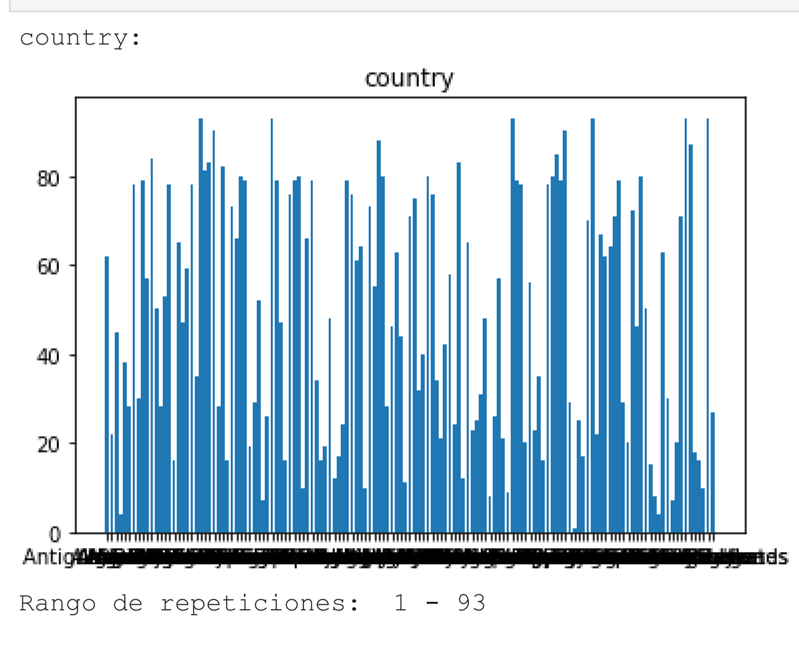
In [12]:
```
plt.plot(df.country,df.people_fully_vaccinated_per_hundred)
plt.title('Vacunaciones completas por país')
plt.show
```

Out[12]: <function matplotlib.pyplot.show(close=None, block=None)>

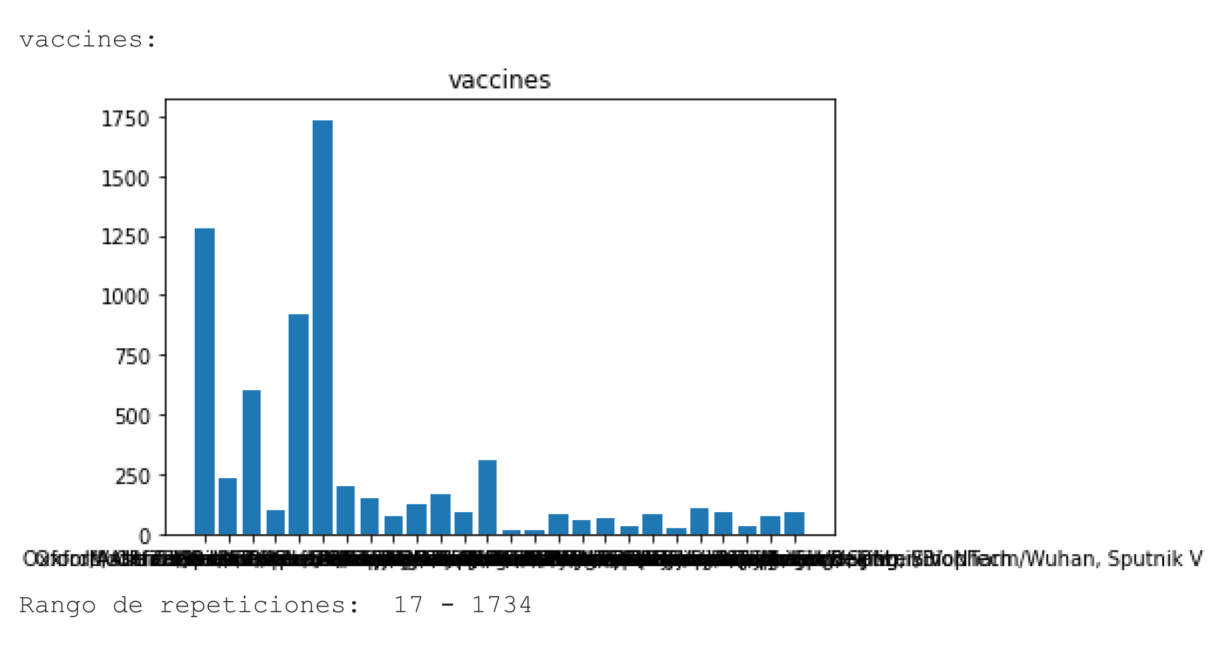


In [13]:
```
#Para cada una de las variables seleccionadas para describir
variables = ["country","vaccines"]

for variable in variables:


    print("\n"+variable+": ")
    repeticiones = {}

    #Contar cuántas veces se repite cada elemento
    for var in df[variable]:
        repeticiones.setdefault(var,0)
        repeticiones[var]+=1

    #Graficar repeticiones
    plt.title(variable)
    plt.bar(repeticiones.keys(),repeticiones.values())
    plt.show()



    #Acomodar diccionario de mayor a menor
    sorted_r = dict(sorted(repeticiones.items(), key=lambda item: item[1],reverse = Tr
    x = 1

    #Publicar rango de repeticiones
    print("Rango de repeticiones: ",sorted_r[list(sorted_r)[-1]],"-",sorted_r[list(so


    print("Primeros lugares: ")

    #Publicar únicamente los cinco primeros elementos, su posición, su nombre y su va
    for key in sorted_r:
        print(x,".- ",key,": ",sorted_r[key])
        x+=1
        if(x>5):
            break
```

```
country:
```




```
Rango de repeticiones:  1 - 93

Primeros lugares:
1 .-  Canada :  93
2 .-  England :  93
3 .-  Northern Ireland :  93
4 .-  Scotland :  93
5 .-  United Kingdom :  93

vaccines:
```



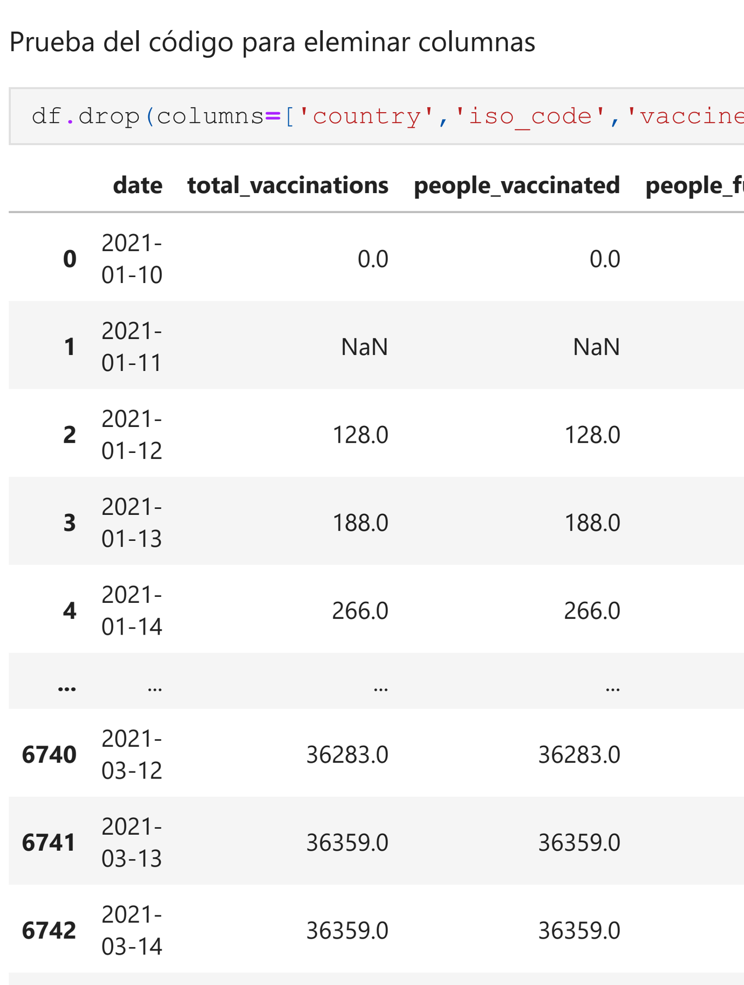

```
Rango de repeticiones:  17 - 1734

Primeros lugares:
1 .-  Moderna, Oxford/AstraZeneca, Pfizer/BioNTech :  1734
2 .-  Pfizer/BioNTech :  1276
3 .-  Oxford/AstraZeneca, Pfizer/BioNTech :  923
4 .-  Oxford/AstraZeneca :  602
5 .-  Moderna, Pfizer/BioNTech :  305
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['country', 'iso_code', 'date', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated',
       'daily_vaccinations_raw', 'daily_vaccinations',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'daily_vaccinations_per_million',
       'vaccines', 'source_name', 'source_website'],
      dtype='object')
```

Generamos el mapa de calor

In [15]:
```
data = (df.total_vaccinations, df.people_fully_vaccinated_per_hundred)
heat_map = sb.heatmap(data)
plt.title('personas vacunadas vs total vacunas')
plt.show()
```



Prueba del código para eliminar columnas

In [32]:
```
df.drop(columns=['country','iso_code','vaccines','source_name','source_website'],axis
```

Out[32]:

	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccina
0	2021-01-10	0.0	0.0	NaN	NaN	
1	2021-01-11	NaN	NaN	NaN	NaN	
2	2021-01-12	128.0	128.0	NaN	NaN	
3	2021-01-13	188.0	188.0	NaN	60.0	
4	2021-01-14	266.0	266.0	NaN	78.0	
...	...	...	...	...	...	
6740	2021-03-12	36283.0	36283.0	NaN	264.0	
6741	2021-03-13	36359.0	36359.0	NaN	76.0	
6742	2021-03-14	36359.0	36359.0	NaN	0.0	
6743	2021-03-15	37660.0	37660.0	NaN	1301.0	
6744	2021-03-16	39550.0	39550.0	NaN	1890.0	

6745 rows × 10 columns

Inicio del algoritmo kmeans, haciendo uso de las columnas sin eliminar

In [35]:
```
X = scale(df.drop(columns=['country','date','iso_code','vaccines','source_name','sourc
y = pd.DataFrame(df.people_fully_vaccinated_per_hundred)

X[0:10,]
```

Out[35]:
```
array([[-0.28419097, -0.31098163,         nan,         nan,         nan,
        -0.55274951, -0.6213969 ,         nan,         nan],
       [        nan,         nan,         nan,         nan, -0.30081677,
                nan,         nan,         nan, -0.61887583],
       [-0.28417505, -0.3109602 ,         nan,         nan, -0.30081677,
        -0.55274951, -0.6213969 ,         nan, -0.61887583],
       [-0.28416759, -0.31095016,         nan, -0.3286807 , -0.30082185,
        -0.55214334, -0.62051274,         nan, -0.61887583],
       [-0.28415789, -0.31093711,         nan, -0.32861136, -0.3008066 ,
        -0.55214334, -0.62051274,         nan, -0.61864053],
       [-0.28415267, -0.31093008,         nan, -0.32875004, -0.30082694,
        -0.55214334, -0.62051274,         nan, -0.61887583],
       [-0.28414508, -0.31091987,         nan, -0.32867685, -0.30082694,
        -0.55214334, -0.62051274,         nan, -0.61887583],
       [-0.2841406 , -0.31091384,         nan, -0.32877315, -0.30084727,
        -0.55214334, -0.62051274,         nan, -0.61934643],
       [-0.28413538, -0.31090681,         nan, -0.32875004, -0.30086252,
        -0.55153718, -0.61962857,         nan, -0.61958173],
       [-0.2841309 , -0.31090079,         nan, -0.32877315, -0.30088286,
        -0.55153718, -0.61962857,         nan, -0.61981703]])
```

Cambiamos todos los NaN por 0's

In [48]:
```
import numpy as np
X[np.isnan(X)] = 0
X
```

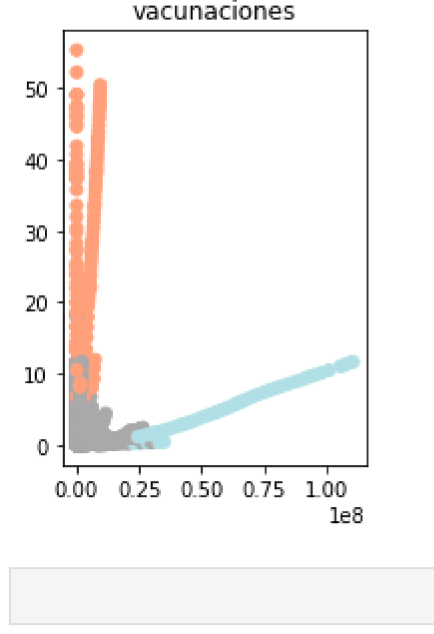
Out[48]:
```
array([[-0.28419097, -0.31098163,  0.        , ..., -0.6213969 ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        , -0.61887583],
       [-0.28417505, -0.3109602 ,  0.        , ..., -0.6213969 ,
         0.        , -0.61887583],
       ...,
       [-0.27966926, -0.30489624,  0.        , ..., -0.60017687,
         0.        , -0.61464044],
       [-0.27950746, -0.30467849,  0.        , ..., -0.5992927 ,
         0.        , -0.61911113],
       [-0.27927242, -0.30436216,  0.        , ..., -0.59752436,
         0.        , -0.61558164]])
```

In [49]:
```
clustering = KMeans(n_clusters = 3, random_state = 5)
clustering.fit(X)
```

Out[49]: KMeans(n_clusters=3, random_state=5)

In [54]:
```
color_theme = np.array(['darkgray', 'lightsalmon', 'powderblue'])
plt.subplot(1,2,1)
plt.scatter(x = df.total_vaccinations, y=df.people_fully_vaccinated_per_hundred, c=co
plt.title("vacunaciones")
```

Out[54]: Text(0.5, 1.0, 'vacunaciones')



In [ ]: