In [1]:

```
# Importar las librerias para el analasis
import pandas as pd
import numpy as np
from datetime import datetime,timedelta
from sklearn.metrics import mean_squared_error
from scipy.optimize import curve_fit
from scipy.optimize import fsolve
from sklearn import linear_model
import matplotlib.pyplot as plt
%matplotlib inline

url = 'C:/Users/ADMINX/Downloads/Compressed/ecuacovid-master/ecuacovid-master/datos_cru

df = pd.read_csv(url)
df
```

Out[1]:

	fecha	dosis_total	primera_dosis	segunda_dosis
0	21/01/2021	0	0	0
1	22/01/2021	108	108	0
2	27/01/2021	2982	2982	0
3	04/02/2021	6228	6228	0
4	17/02/2021	8190	6228	1962
5	24/02/2021	24492	20784	3708
6	01/03/2021	42114	35886	6228
7	04/03/2021	59316	53088	6228
8	05/03/2021	71148	64920	6228
9	08/03/2021	74472	68244	6228
10	09/03/2021	75258	69030	6228
11	11/03/2021	95915	89687	6228
12	12/03/2021	123176	116948	6228
13	13/03/2021	139359	119222	20137
14	15/03/2021	141191	121054	20137
15	21/03/2021	178970	140765	38205
16	23/03/2021	182261	143614	38647
17	24/03/2021	191179	152526	38653
18	26/03/2021	230770	172413	58357
19	27/03/2021	235000	174642	60358
20	29/03/2021	244866	182329	62537
21	01/04/2021	283106	204902	78204
22	04/04/2021	301069	211720	89349
23	05/04/2021	335093	228504	106589

	fecha	dosis_total	primera_dosis	segunda_dosis
24	06/04/2021	356783	244159	112624
25	08/04/2021	363255	250631	112624
26	14/04/2021	480962	338180	142782
27	15/04/2021	485132	338180	146952
28	16/04/2021	514151	354019	160132
29	17/04/2021	545132	377199	167933
30	18/04/2021	554369	384093	170276
31	19/04/2021	577711	401871	175840
32	20/04/2021	601229	421937	179292
33	21/04/2021	643702	457403	186299
34	22/04/2021	675510	486524	188986
35	23/04/2021	711204	514854	196350
36	24/04/2021	732717	532367	200350
37	25/04/2021	743937	541420	202517
38	26/04/2021	765489	555265	210224
39	27/04/2021	816175	595699	220476
40	28/04/2021	861393	633421	227972
41	29/04/2021	920865	691000	229865
42	30/04/2021	987452	748021	239431
43	01/05/2021	1036794	791822	244972
44	02/05/2021	1067472	821960	245512
45	04/05/2021	1141262	889218	252044
46	05/05/2021	1182085	924539	257546
47	06/05/2021	1215676	953238	262438
48	07/05/2021	1245822	981620	264202

Generar graficas y reportes del total de personas vacunadas.

In [25]:

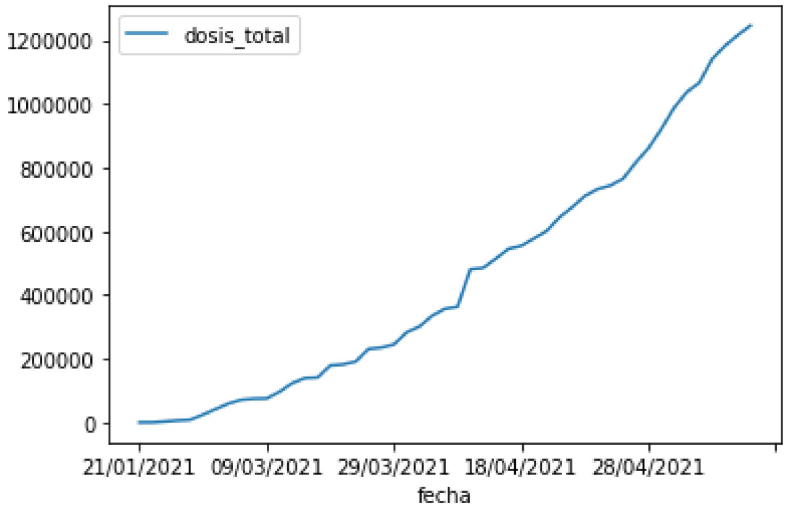
```
df_t = pd.read_csv(url,header = None)
FMT = '%Y-%m-%d'
df_t.columns =['fecha','dosis_total','primera_dosis', 'segunda_dosis']
date = df_t['fecha'][1:]
filtro = df_t['dosis_total'][1:] # Filtro los datos que se empezo a tener casos
#Obtenemos la mediana
media = filtro.mean()
mediana = filtro.median()
print(mediana)
print(media)
df.plot(x ='fecha', y='dosis_total')

```

```
356783.0
2.2101686181263315e+275
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20c7d25f248>
```



Generar grafico de pie por fabricante de la vacuna.

In [28]:

```
1 url = 'C:/Users/ADMINX/Downloads/Compressed/ecuacovid-master/ecuacovid-master/datos_cru
2 df = pd.read_csv(url)
3 df
```

Out[28]:

	vaccine	total	arrived_at
0	Pfizer/BioNTech	8190	20/01/2021
1	Pfizer/BioNTech	16380	17/02/2021
2	Pfizer/BioNTech	17550	24/02/2021
3	Pfizer/BioNTech	31590	03/03/2021
4	Sinovac	20000	06/03/2021
5	Pfizer/BioNTech	73710	10/03/2021
6	Oxford/AstraZeneca	84000	17/03/2021
7	Pfizer/BioNTech	62010	17/03/2021
8	Pfizer/BioNTech	65520	24/03/2021
9	Pfizer/BioNTech	66690	31/03/2021
10	Pfizer/BioNTech	53820	05/04/2021
11	Sinovac	300000	07/04/2021
12	Sinovac	700000	10/04/2021
13	Pfizer/BioNTech	53820	14/04/2021
14	Pfizer/BioNTech	54990	21/04/2021
15	Oxford/AstraZeneca	336000	24/04/2021
16	Pfizer/BioNTech	54990	28/04/2021
17	Pfizer/BioNTech	100620	04/05/2021

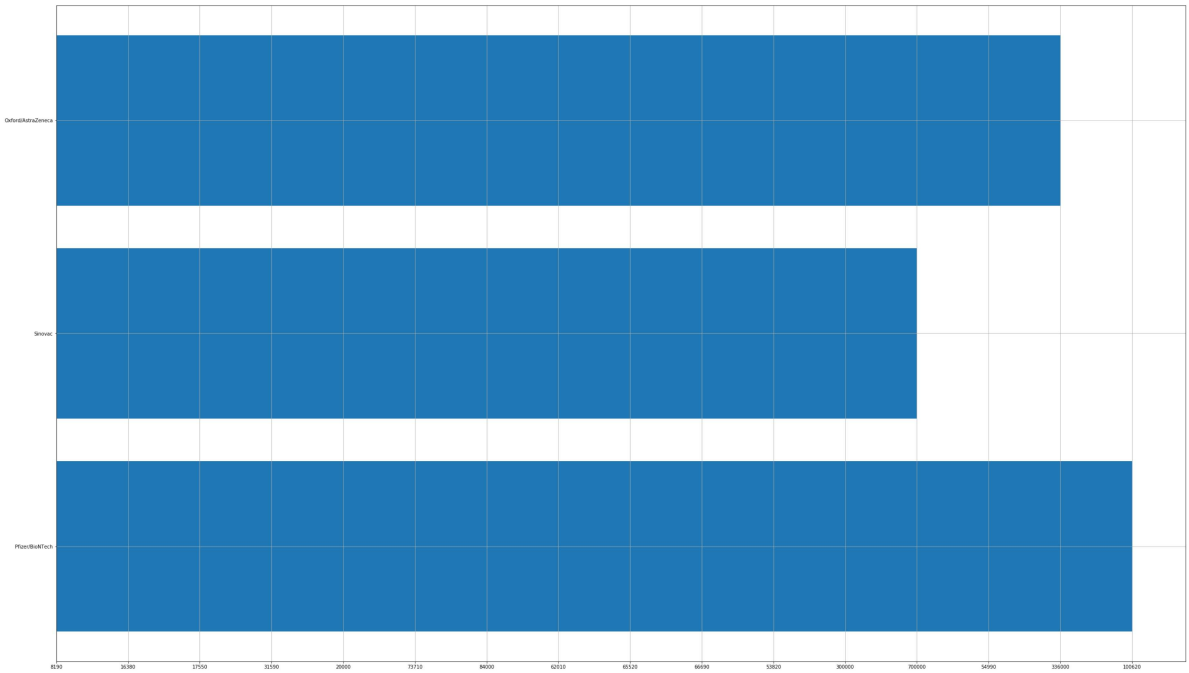
In [33]:

```
df_fa = pd.read_csv(url, header = None)
df_fa.columns = ['vaccine', 'total', 'arrived_at']
a = np.array(list(set(df_fa['vaccine'][1:])))
print(a)

ox = 0
si = 0
pf = 0
for i, j in zip(df_fa['vaccine'][1:], df_fa['total'][1:]):
  if i == a[0]:
    si = si + + int(j)
  elif i == a[1]:
    pf = pf + + int(j)
  elif i == a[2]:
    ox = ox + int(j)

b = np.array([si, pf, ox])
print(b)
plt.barh(df_fa['vaccine'][1:], df_fa['total'][1:])
plt.gcf().set_size_inches(42, 25)
plt.grid()
plt.show()
```

```
['Oxford/AstraZeneca' 'Sinovac' 'Pfizer/BioNTech']
[ 420000 1020000  659880]
```

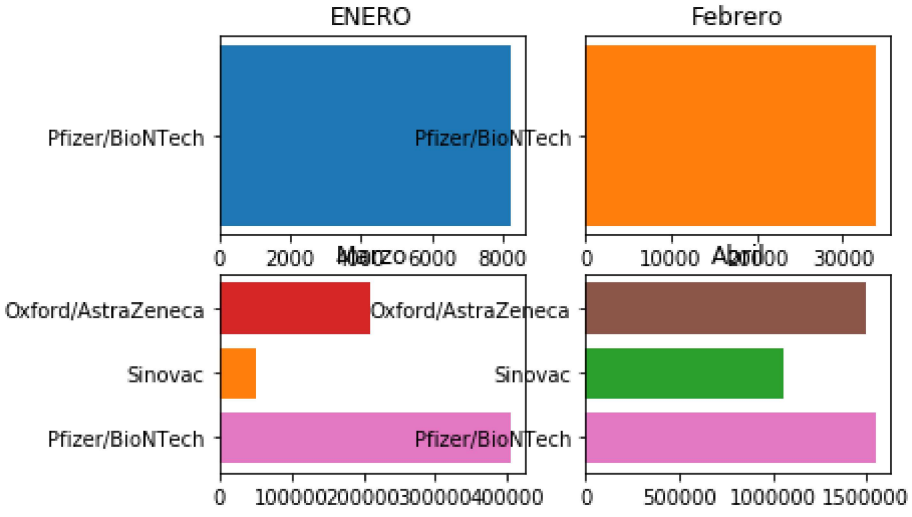


Generar histogramas de vacunas por mes de llega y fabricante.

In [177]:

```
from ipykernel import kernelapp as app
lista1=[]
fig1 = plt.subplot(2, 2, 1)
fig2 = plt.subplot(2, 2, 2)
fig3 = plt.subplot(2, 2, 3)
fig4 = plt.subplot(2, 2, 4)
for i in range(1, 32):
  lista1.append(0)
con = 0
con1 = 0
con2=0
con3=0
con4=0

for i, j, k in zip(df_fa['vaccine'][1:], df_fa['total'][1:], df_fa['arrived_at'][1:]):
  fecha_dt = datetime.strptime(k, '%d/%m/%Y')
  if((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month == 1)):
      con =con+int(j)
      fig1.barh(i,con)
      fig1.set_title('ENERO')
  elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month == 2)):
      con1 =con1+int(j)
      fig2.barh(i,con1)
      fig2.set_title('Febrero')
  elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month == 3)):
      con2 =con2+int(j)
      fig3.barh(i,con2)
      fig3.set_title('Marzo')
  elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month == 4)):
      con3 =con3+int(j)
      fig4.barh(i,con3)
      fig4.set_title('Abril')
#   elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month == 5)):
#       con4 =con4+int(j)
#       plt.barh(i,con4)
#       plt.title('Mayo')
```



Generar un reporte parametrizado que pueda ingresar los datos de las fechas inicio y fin para obtener la información de las graficas vistas en el primer punto.

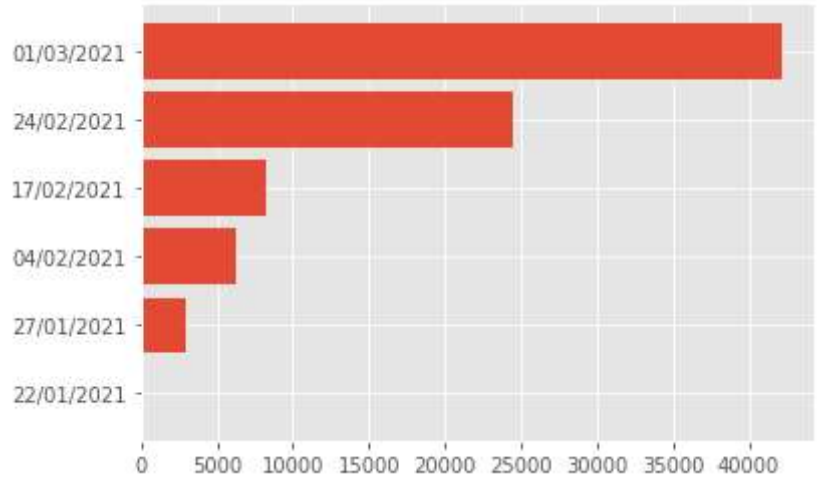
In [24]:

```
df_planva = pd.read_csv(url)
lisx=[]
lisy=[]
df_planva.columns =['fecha','dosis_total','primera_dosis', 'segunda_dosis']
#SE INGRESA LAS FECHAS INICIO Y FIN
feini=input("ingrese fecha de inicio: ")
fefin=input("ingrese fecha de fin: ")
for d,i, j, k in zip(df_planva['fecha'][1:],df_planva['dosis_total'][1:], df_planva['pr
  fecha_dt = datetime.strptime(d, '%d/%m/%Y')
  if(datetime.strptime(feini,'%d/%m/%Y')<=datetime.strptime(d, '%d/%m/%Y') and datetime
    lisx.append(d)
    lisy.append(i)
plt.barh(lisx,lisy)
```

```
ingrese fecha de inicio: 01/01/2021
ingrese fecha de fin: 01/03/2021
```

Out[24]:

```
<BarContainer object of 6 artists>
```



In [ ]:

```
1
```

Generar un modelo matemático de predicción para regresión lineal, exponencial, polinómico y logarítmico, del procesos de vacunación en base al numero actual de vacunados (1 y 2 dosis) y a la llegada de nuevas vacunas

In [10]:

```
#SE APLICA UN MODELO MATEMATICO DE REGRESION LINEAL

# Tratamiento de datos
# ==============================================================================
import pandas as pd
import numpy as np

# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y modelado
# ==============================================================================
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Configuración matplotlib
# ==============================================================================
plt.rcParams['image.cmap'] = "bwr"
#plt.rcParams['figure.dpi'] = "100"
plt.rcParams['savefig.bbox'] = "tight"
style.use('ggplot') or plt.style.use('ggplot')

# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
url = 'C:/Users/ADMINX/Downloads/Compressed/ecuacovid-master/ecuacovid-master/datos_cru
df_dosis = pd.read_csv(url)
df_dosis

```

Out[10]:

	fecha	dosis_total	primera_dosis	segunda_dosis
0	21/01/2021	0	0	0
1	22/01/2021	108	108	0
2	27/01/2021	2982	2982	0
3	04/02/2021	6228	6228	0
4	17/02/2021	8190	6228	1962
5	24/02/2021	24492	20784	3708
6	01/03/2021	42114	35886	6228
7	04/03/2021	59316	53088	6228
8	05/03/2021	71148	64920	6228

	fecha	dosis_total	primera_dosis	segunda_dosis
9	08/03/2021	74472	68244	6228
10	09/03/2021	75258	69030	6228
11	11/03/2021	95915	89687	6228
12	12/03/2021	123176	116948	6228
13	13/03/2021	139359	119222	20137
14	15/03/2021	141191	121054	20137
15	21/03/2021	178970	140765	38205
16	23/03/2021	182261	143614	38647
17	24/03/2021	191179	152526	38653
18	26/03/2021	230770	172413	58357
19	27/03/2021	235000	174642	60358
20	29/03/2021	244866	182329	62537
21	01/04/2021	283106	204902	78204
22	04/04/2021	301069	211720	89349
23	05/04/2021	335093	228504	106589
24	06/04/2021	356783	244159	112624
25	08/04/2021	363255	250631	112624
26	14/04/2021	480962	338180	142782
27	15/04/2021	485132	338180	146952
28	16/04/2021	514151	354019	160132
29	17/04/2021	545132	377199	167933
30	18/04/2021	554369	384093	170276
31	19/04/2021	577711	401871	175840
32	20/04/2021	601229	421937	179292
33	21/04/2021	643702	457403	186299
34	22/04/2021	675510	486524	188986
35	23/04/2021	711204	514854	196350
36	24/04/2021	732717	532367	200350
37	25/04/2021	743937	541420	202517
38	26/04/2021	765489	555265	210224
39	27/04/2021	816175	595699	220476
40	28/04/2021	861393	633421	227972
41	29/04/2021	920865	691000	229865
42	30/04/2021	987452	748021	239431
43	01/05/2021	1036794	791822	244972
44	02/05/2021	1067472	821960	245512
45	04/05/2021	1141262	889218	252044
46	05/05/2021	1182085	924539	257546
47	06/05/2021	1215676	953238	262438

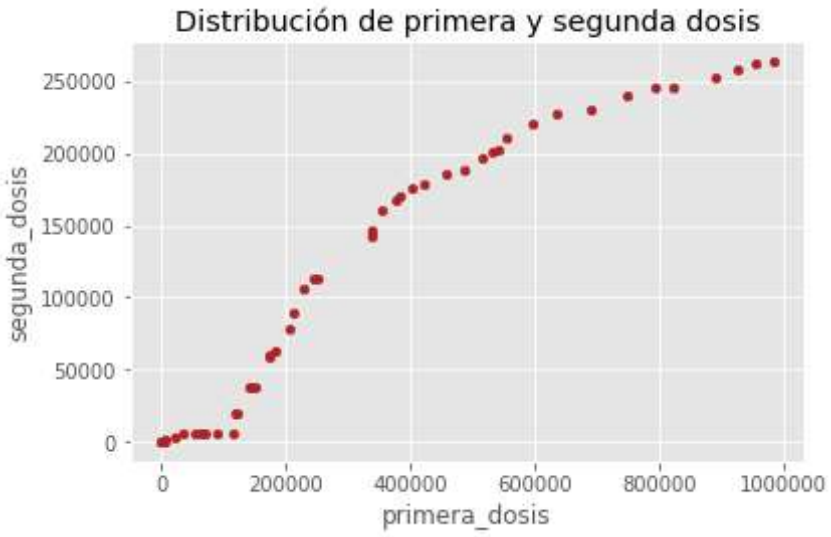
	fecha	dosis_total	primera_dosis	segunda_dosis
48	07/05/2021	1245822	981620	264202

In [13]:

```
# Gráfico de Dosis
# ================================================================================
fig, ax = plt.subplots(figsize=(6, 3.84))

# plt.barh(df_dosis['primera_dosis'][1:], df_fa['segunda_dosis'][1:])
df_dosis.plot(
    x    = 'primera_dosis',
    y    = 'segunda_dosis',
    c    = 'firebrick',
    kind = "scatter",
    ax   = ax
)
ax.set_title('Distribución de primera y segunda dosis');
# Correlación lineal entre las dos variables
# ================================================================================
corr_test = pearsonr(x = df_dosis['primera_dosis'], y =  df_dosis['segunda_dosis'])
print("Coeficiente de correlación : ", corr_test[0])
print("P-value: ", corr_test[1])
```

```
Coeficiente de correlación de Pearson:  0.9605961905061328
P-value:  8.842415110079586e-28
```



In [17]:

```
# División de los datos en train y test
# ==============================================================================
X = df_dosis[['primera_dosis']]
y = df_dosis['segunda_dosis']

X_train, X_test, y_train, y_test = train_test_split(
                                        X.values.reshape(-1,1),
                                        y.values.reshape(-1,1),
                                        train_size   = 0.8,
                                        random_state = 1234,
                                        shuffle      = True
                                    )

# Creación del modelo
# ==============================================================================
modelo = LinearRegression()
modelo.fit(X = X_train.reshape(-1, 1), y = y_train)

```

Out[17]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [19]:

```

# Información del modelo
# ==============================================================================
print("Intercept:", modelo.intercept_)
print("Coeficiente:", list(zip(X.columns, modelo.coef_.flatten(), )))
print("Coeficiente de determinación R^2:", modelo.score(X, y))


# Error de test del modelo
# ==============================================================================
predicciones = modelo.predict(X = X_test)
print(predicciones[0:3,])

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print("")
print(f"El error (rmse) de test es: {rmse}")

```

```
Intercept: [8292.96678152]
Coeficiente: [('primera_dosis', 0.3230412116667086)]
Coeficiente de determinación R^2: 0.9215533722399717
[[ 29264.80224293]
 [325396.6809778 ]
 [ 47398.39761863]]

El error (rmse) de test es: 30974.524313787628
```

In [20]:

```
# Creación del modelo utilizando matrices como en scikitlearn
# ==============================================================================
# A la matriz de predictores se le tiene que añadir una columna de 1s para el intercept
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.925
Model:                            OLS   Adj. R-squared:                  0.923
Method:                 Least Squares   F-statistic:                     457.5
Date:                Tue, 11 May 2021   Prob (F-statistic):           2.00e-22
Time:                        17:57:49   Log-Likelihood:                -450.45
No. Observations:                  39   AIC:                             904.9
Df Residuals:                      37   BIC:                             908.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       8292.9668   6450.578      1.286      0.207   -4777.146    2.14e+04
x1             0.3230      0.015     21.390      0.000       0.292       0.354
==============================================================================
Omnibus:                        3.205   Durban-Watson:                   1.939
Prob(Omnibus):                  0.201   Jarque-Bera (JB):                1.943
Skew:                          -0.309   Prob(JB):                        0.378
Kurtosis:                       2.098   Cond. No.                     6.67e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.67e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [21]:

```
# Intervalos de confianza para los coeficientes del modelo
# ============================================================================
modelo.conf_int(alpha=0.05)
```

Out[21]:

```
array([[-4.77714629e+03,  2.13630799e+04],
       [ 2.92440592e-01,  3.53641831e-01]])
```

In [22]:

```
# Predicciones con intervalo de confianza del 95%
# ============================================================================
predicciones = modelo.get_prediction(exog = X_train).summary_frame(alpha=0.05)
predicciones.head(4)
```

Out[22]:

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	180269.447513	5152.495548	169829.499867	190709.395159	127001.498896	233537.396130
1	264084.105086	8128.113381	247614.983015	280553.227156	209314.461808	318853.748364
2	57565.150632	4907.175337	47622.268950	67508.032315	4392.387500	110737.913765
3	30592.501623	5688.744669	19066.010051	42118.993195	-22899.010330	84084.013575

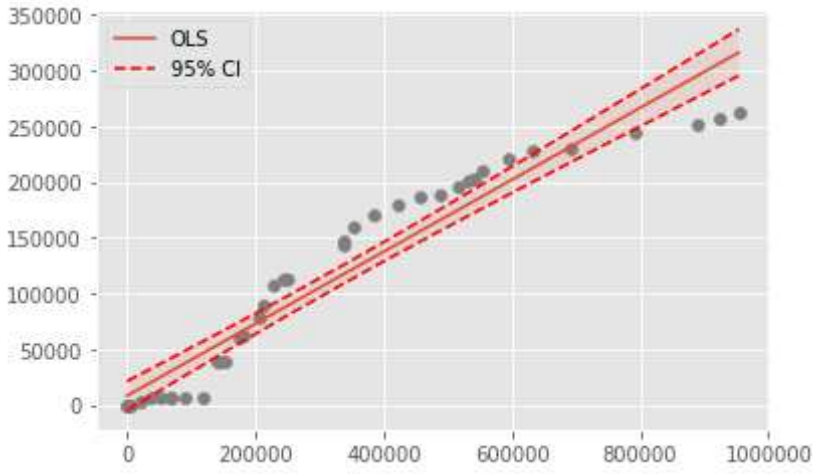
In [23]:

```
# Predicciones con intervalo de confianza del 95%
# ==============================================================================
predicciones = modelo.get_prediction(exog = X_train).summary_frame(alpha=0.05)
predicciones['x'] = X_train[:, 1]
predicciones['y'] = y_train
predicciones = predicciones.sort_values('x')

# Gráfico del modelo
# ==============================================================================
fig, ax = plt.subplots(figsize=(6, 3.84))

ax.scatter(predicciones['x'], predicciones['y'], marker='o', color = "gray")
ax.plot(predicciones['x'], predicciones["mean"], linestyle='-', label="OLS")
ax.plot(predicciones['x'], predicciones["mean_ci_lower"], linestyle='--', color='red',
ax.plot(predicciones['x'], predicciones["mean_ci_upper"], linestyle='--', color='red')
ax.fill_between(predicciones['x'], predicciones["mean_ci_lower"], predicciones["mean_ci
ax.legend();

```



Desarrollar y generar un proceso de comparación con al menos cuatro países (2. Latinoamérica, 1. E.E.U.U./Canada, 1. Europa

In [5]:

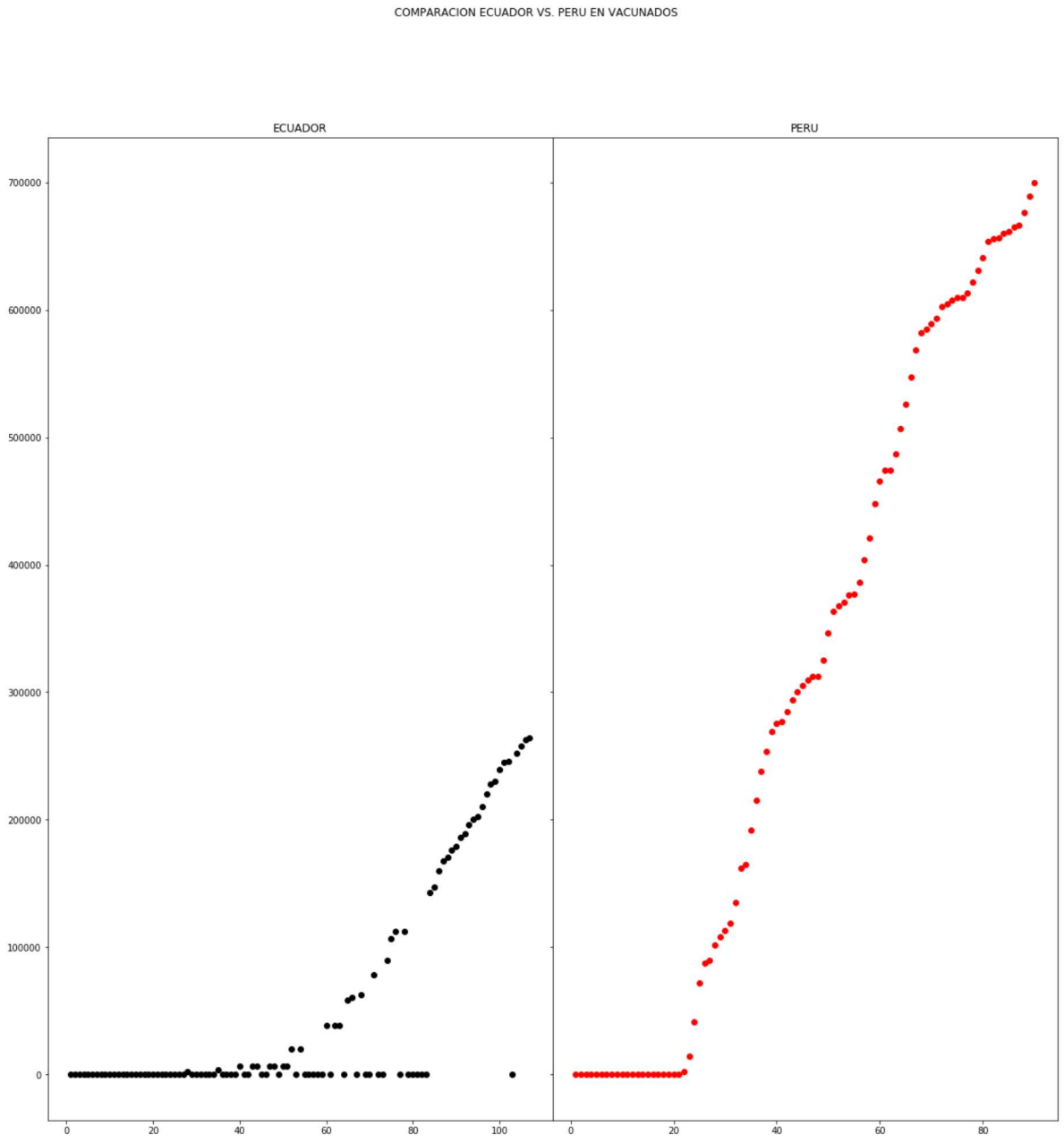
```
url_v = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccin
df_vacum = pd.read_csv(url_v, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinations', 'people_vacci
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum['people_fully_vacci
#   print(i)
  if i == 'Ecuador':
    lista_fechas.append(j)
    lista_tot.append(k)
  elif i == 'Peru':
    lista_fechasbr.append(j)
    lista_totbr.append(k)
dat = np.array(lista_fechas)
tot = np.array(lista_tot, dtype='float')
x = np.arange(1, len(tot) + 1, 1)
dat_br = np.array(lista_fechasbr)
tot_br = np.array(lista_totbr, dtype='float')
xbr = np.arange(1, len(tot_br) + 1, 1)
from sklearn import linear_model
print("-----------------ECUADOR--------------")
regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1), tot)
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("----------------PERU---------------")
regrbr = linear_model.LinearRegression()
regrbr.fit(np.array(xbr).reshape(-1, 1), tot_br)
print('Coefficients: \n', regrbr.coef_)
print('Independent term: \n', regrbr.intercept_)
#SE GRAFICA
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                          gridspec_kw={'hspace': 0, 'wspace': 0}, figsize=(20,20))
fig.suptitle('COMPARACION ECUADOR VS. PERU EN VACUNADOS')
ax1.scatter(x, tot, color='black')
ax1.set_title('ECUADOR')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('PERU')

```

```
-----------------ECUADOR--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
----------------PERU---------------
Coefficients:
 [9405.52787587]
Independent term:
 -118419.22946317098
```

Out[5]:

```
Text(0.5, 1.0, 'PERU')
```



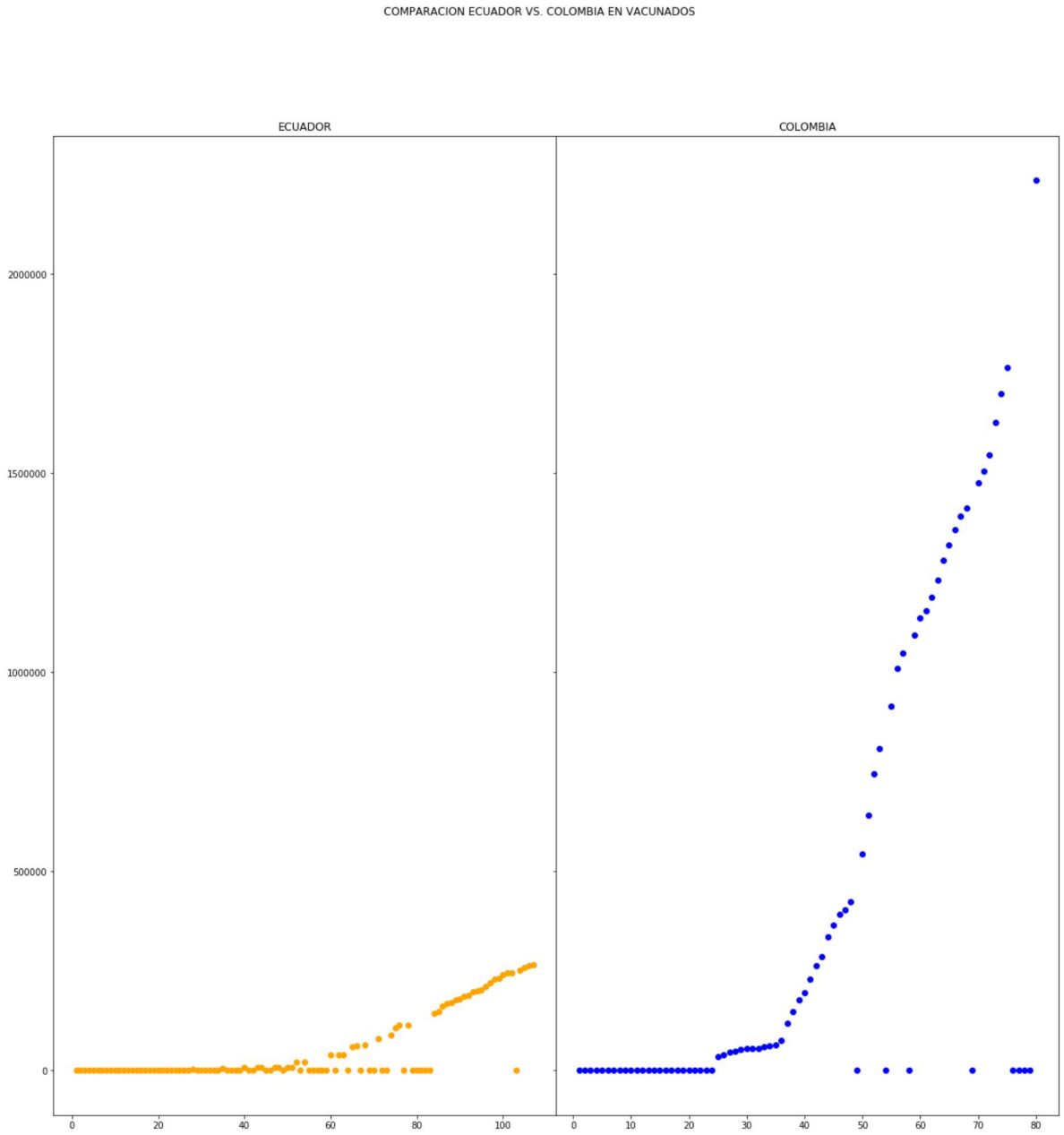
In [6]:

```
url_v = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccin
df_vacum = pd.read_csv(url_v, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinations', 'people_vacci
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum['people_fully_vacci
#   print(i)
  if i == 'Ecuador':
    lista_fechas.append(j)
    lista_tot.append(k)
  elif i == 'Colombia':
    lista_fechasbr.append(j)
    lista_totbr.append(k)
dat = np.array(lista_fechas)
tot = np.array(lista_tot, dtype='float')
x = np.arange(1, len(tot) + 1, 1)
dat_br = np.array(lista_fechasbr)
tot_br = np.array(lista_totbr, dtype='float')
xbr = np.arange(1, len(tot_br) + 1, 1)
from sklearn import linear_model
print("-----------------ECUADOR--------------")
regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1), tot)
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("----------------COLOMBIA---------------")
regrbr = linear_model.LinearRegression()
regrbr.fit(np.array(xbr).reshape(-1, 1), tot_br)
print('Coefficients: \n', regrbr.coef_)
print('Independent term: \n', regrbr.intercept_)
#SE GRAFICA
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                        gridspec_kw={'hspace': 0, 'wspace': 0}, figsize=(20,20))
fig.suptitle('COMPARACION ECUADOR VS. COLOMBIA EN VACUNADOS')
ax1.scatter(x, tot, color='orange')
ax1.set_title('ECUADOR')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='blue')
ax2.set_title('COLOMBIA')
```

```
-----------------ECUADOR--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
----------------COLOMBIA---------------
Coefficients:
 [18072.47528129]
Independent term:
 -305725.71139240486
```

Out[6]:

```
Text(0.5, 1.0, 'COLOMBIA')
```


COMPARACION ECUADOR VS. COLOMBIA EN VACUNADOS
ECUADOR
COLOMBIA
2000000
1500000
1000000
500000
0
0
20
40
60
80
100
0
10
20
30
40
50
60
70
80

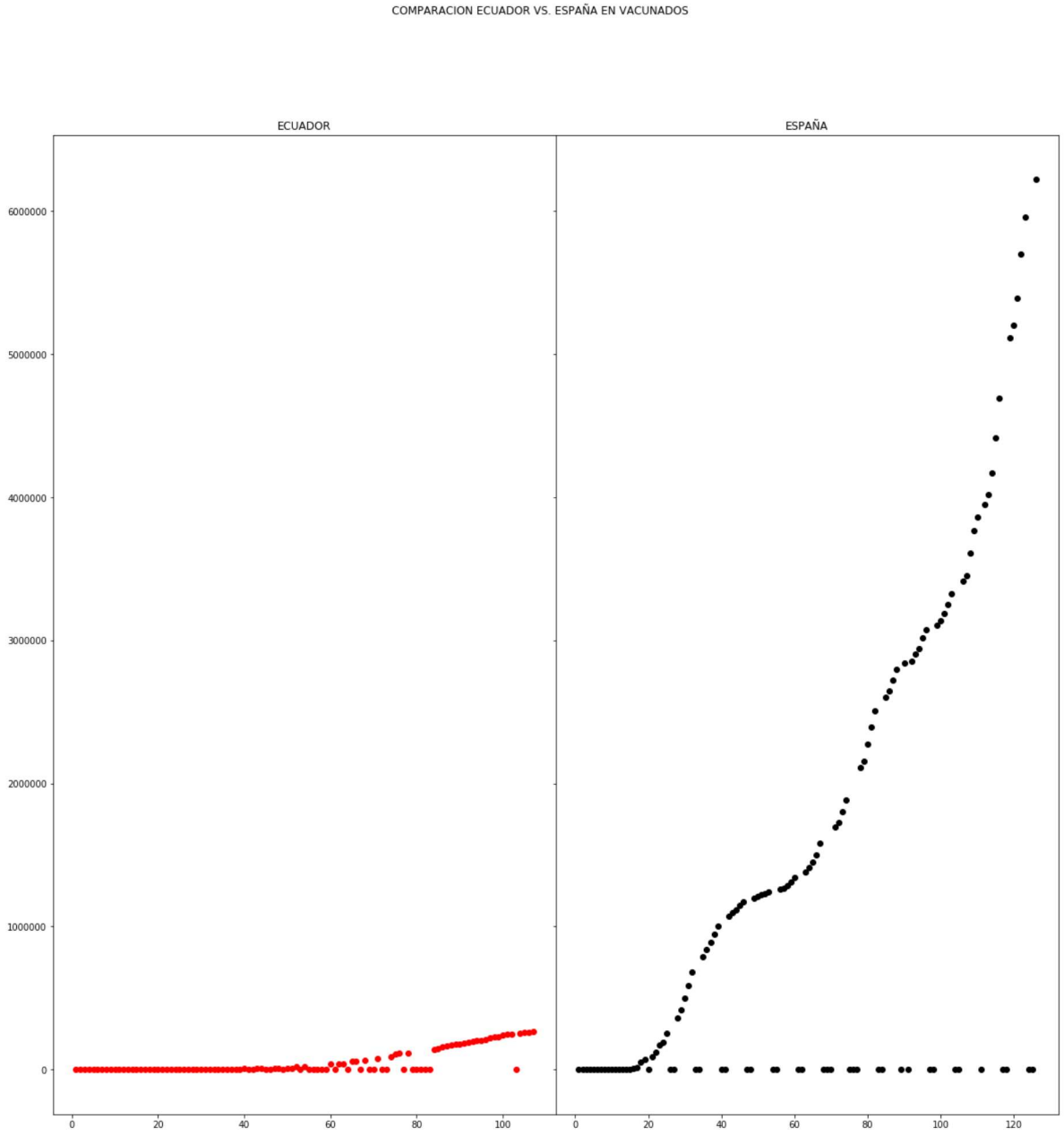
In [7]:

```
url_v = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccin
df_vacum = pd.read_csv(url_v, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinations', 'people_vacci
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum['people_fully_vacci
#   print(i)
  if i == 'Ecuador':
    lista_fechas.append(j)
    lista_tot.append(k)
  elif i == 'Spain':
    lista_fechasbr.append(j)
    lista_totbr.append(k)
dat = np.array(lista_fechas)
tot = np.array(lista_tot, dtype='float')
x = np.arange(1, len(tot) + 1, 1)
dat_br = np.array(lista_fechasbr)
tot_br = np.array(lista_totbr, dtype='float')
xbr = np.arange(1, len(tot_br) + 1, 1)
from sklearn import linear_model
print("-----------------ECUADOR--------------")
regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1), tot)
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("----------------ESPAÑA---------------")
regrbr = linear_model.LinearRegression()
regrbr.fit(np.array(xbr).reshape(-1, 1), tot_br)
print('Coefficients: \n', regrbr.coef_)
print('Independent term: \n', regrbr.intercept_)
#SE GRAFICA
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                        gridspec_kw={'hspace': 0, 'wspace': 0}, figsize=(20,20))
fig.suptitle('COMPARACION ECUADOR VS. ESPAÑA EN VACUNADOS')
ax1.scatter(x, tot, color='red')
ax1.set_title('ECUADOR')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='black')
ax2.set_title('ESPAÑA')
```

```
-----------------ECUADOR--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
----------------ESPAÑA---------------
Coefficients:
 [30146.30433596]
Independent term:
 -570057.5475555563
```

Out[7]:

```
Text(0.5, 1.0, 'ESPAÑA')
```


COMPARACION ECUADOR VS. ESPAÑA EN VACUNADOS
ECUADOR
ESPAÑA
6000000
5000000
4000000
3000000
2000000
1000000
0
0
20
40
60
80
100
120

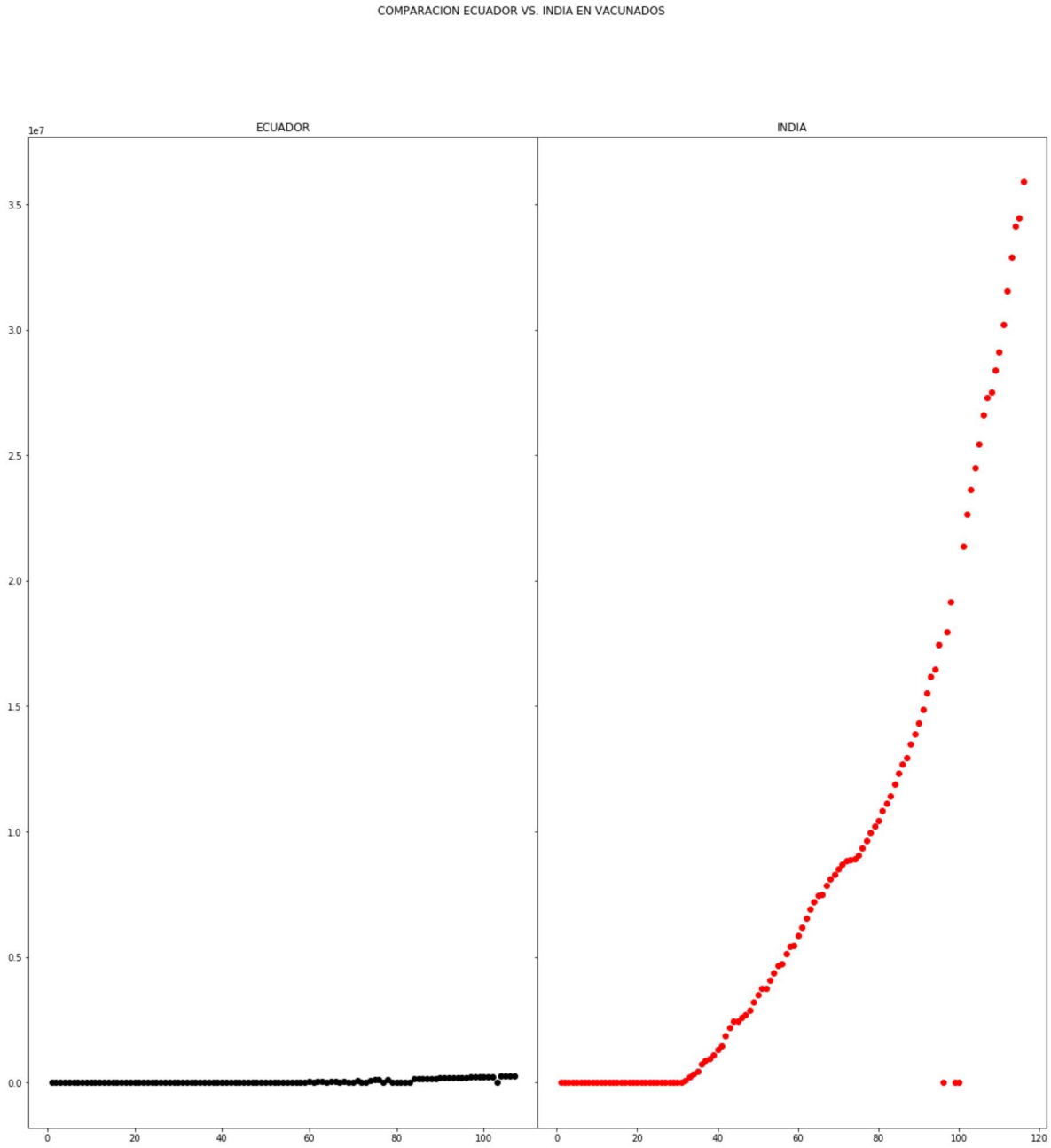
In [8]:

```
url_v = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccin
df_vacum = pd.read_csv(url_v, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinations', 'people_vacci
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum['people_fully_vacci
#   print(i)
  if i == 'Ecuador':
    lista_fechas.append(j)
    lista_tot.append(k)
  elif i == 'India':
    lista_fechasbr.append(j)
    lista_totbr.append(k)
dat = np.array(lista_fechas)
tot = np.array(lista_tot, dtype='float')
x = np.arange(1, len(tot) + 1, 1)
dat_br = np.array(lista_fechasbr)
tot_br = np.array(lista_totbr, dtype='float')
xbr = np.arange(1, len(tot_br) + 1, 1)
from sklearn import linear_model
print("-----------------ECUADOR--------------")
regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1), tot)
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("----------------INDIA---------------")
regrbr = linear_model.LinearRegression()
regrbr.fit(np.array(xbr).reshape(-1, 1), tot_br)
print('Coefficients: \n', regrbr.coef_)
print('Independent term: \n', regrbr.intercept_)
#SE GRAFICA
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                          gridspec_kw={'hspace': 0, 'wspace': 0}, figsize=(20,20))
fig.suptitle('COMPARACION ECUADOR VS. INDIA EN VACUNADOS')
ax1.scatter(x, tot, color='black')
ax1.set_title('ECUADOR')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('INDIA')
```

```
-----------------ECUADOR--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
----------------INDIA---------------
Coefficients:
 [248569.75313497]
Independent term:
 -6389449.032533734
```

Out[8]:

```
Text(0.5, 1.0, 'INDIA')
```



Identificar cual es la fecha tentativa en la que todos los Ecuatorianos podrán ser vacunados con las dos dosis

In [ ]:

```
1
```

Cual tiene una mejor predicción.

Ventajas: Aplicar una lectura de manera grafica y prediciendo datos que se obtiene para poder clasificar bien los datos.

Desventajas de los modelos: Usar los datos para poder aplicar un buen sistema de simulacion, es complicado si no se tiene los datos necesarios y la logica de estudio.
Conclusiones: Es bueno poder aplicar prediciones con datos en los cuales importe mucho ya que es horientado a la salud. Recomendaciones:Realizar mas ejemplos como sistemas donde se aplice la lectura de datos reales

In [ ]:

```
1
```