Roberto Pacho

jpachom1@est.ups.edu.ec (mailto:jpachom1@est.ups.edu.ec)

- Enunciado:

• Diseñe y desarrolle un modelo y/o script que permita simular el siguiente caso real: ◦ Se tiene los datos del ecuador (https://github.com/andrab/ecuacovid/tree/master/datos_crudos (https://github.com/andrab/ecuacovid/tree/master/datos_crudos)). En base a ello obtener los siguientes modelos:

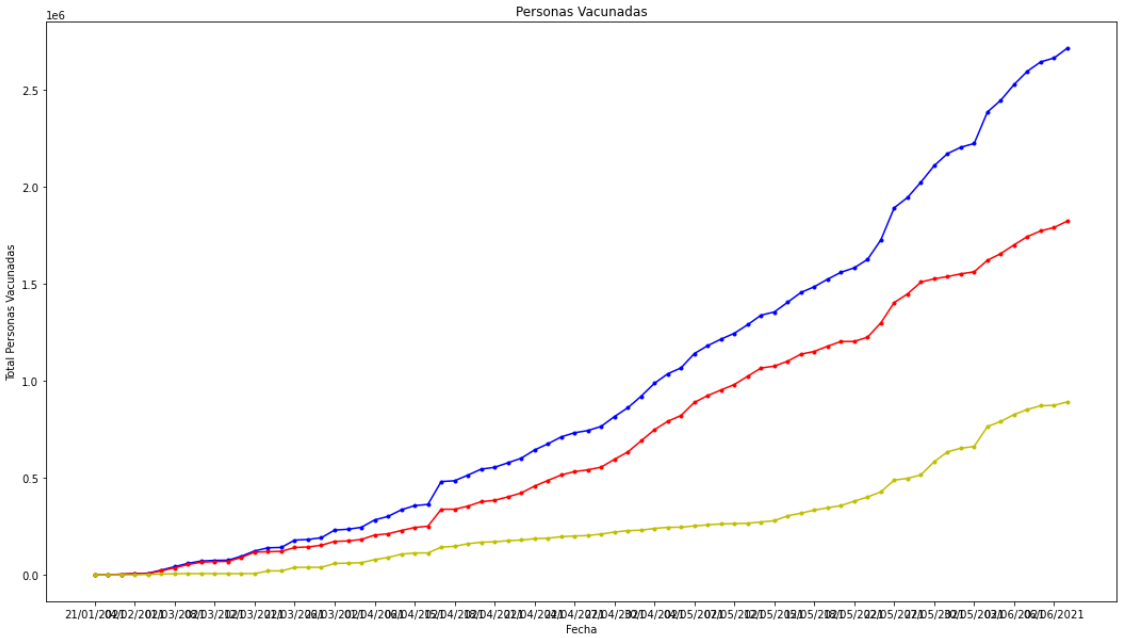
Generar gráficas para entender y procesar los datos:

Generar gráficas y reportes del total de personas vacunadas.

In [11]:
```
#importar las librerias necesarias
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from datetime import datetime
import datetime
import matplotlib.pyplot as plt
from ipykernel import kernelapp as app
import datetime
from sklearn import linear_model
from scipy.optimize import curve_fit
```

In [3]:
```
datos_vacu=pd.read_csv('https://raw.githubusercontent.com/andrab/ecu

plt.figure(figsize=(18,10))

plt.title('Personas Vacunadas')

plt.plot(datos_vacu.fecha, datos_vacu.dosis_total, 'b.-')
plt.plot(datos_vacu.primera_dosis, 'r.-')
plt.plot(datos_vacu.segunda_dosis, 'y.-')

plt.xticks(datos_vacu.fecha[::3].tolist())

plt.xlabel('Fecha')
plt.ylabel('Total Personas Vacunadas')
```

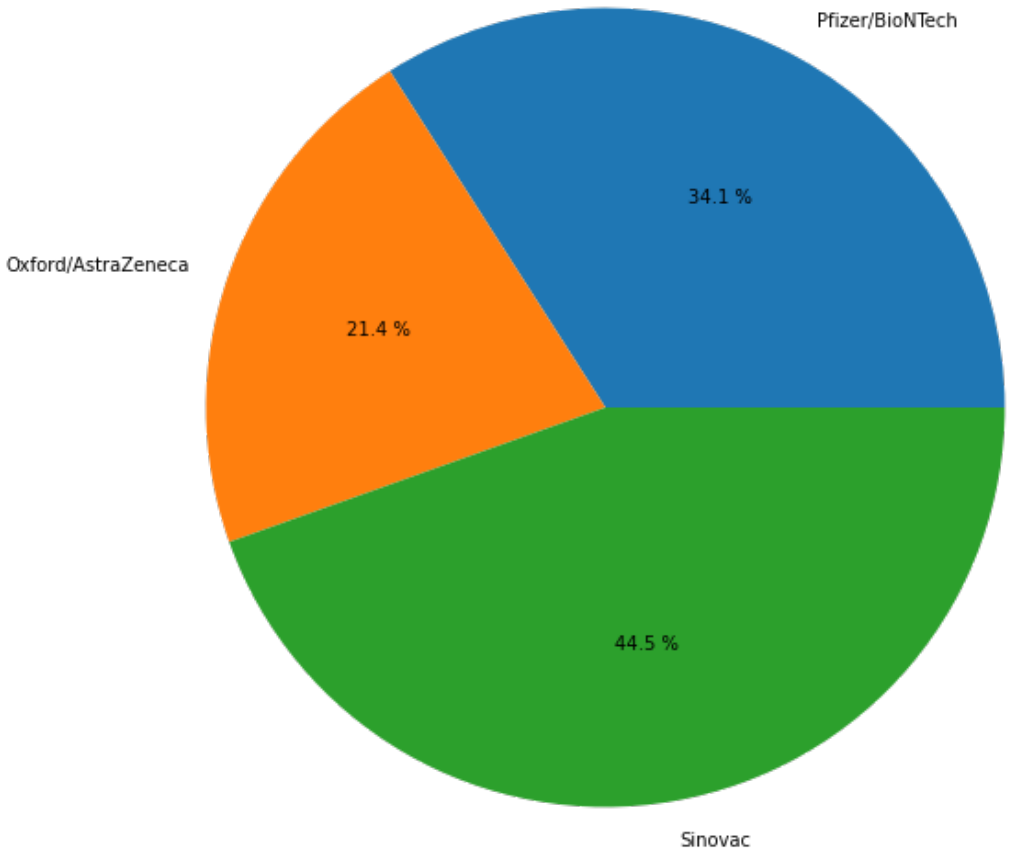
```
plt.show()
```



Generar gráfico de pie por fabricante de la vacuna.

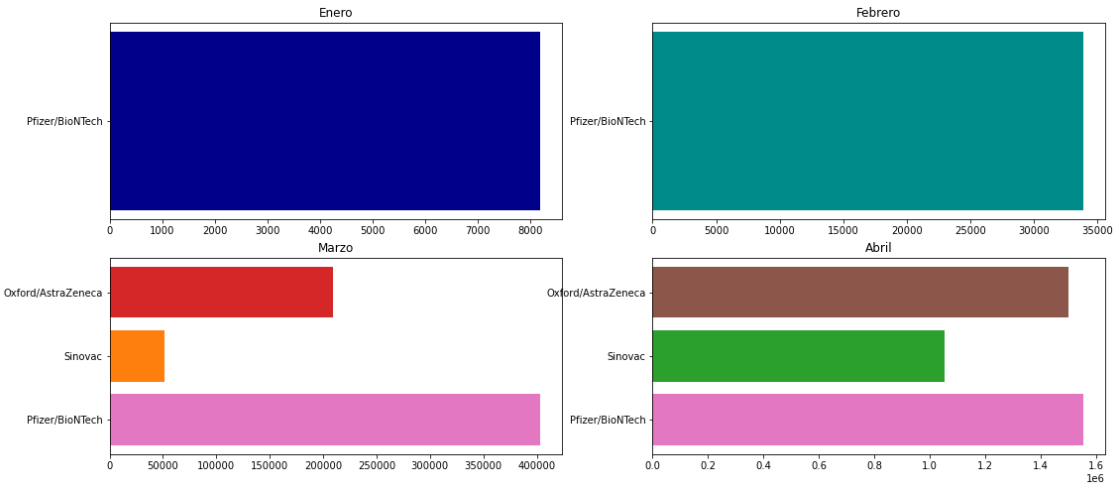
In [7]:
```
datos_fabri=pd.read_csv('https://raw.githubusercontent.com/andrab/e
header=None)
datos_fabri.columns = ['vaccine', 'total', 'arrived_at','contract']
a = np.array(list(set(datos_fabri['vaccine'][1:])))
print(a)

oxford = 0
sino = 0
pfizer = 0
for i, j in zip(datos_fabri['vaccine'][1:], datos_fabri['total'][1:
    if i == a[0]:

        sino = sino + + int(j)
    elif i == a[1]:
        pfizer = pfizer + + int(j)
    elif i == a[2]:
        oxford = oxford + int(j)

b = np.array([sino, pfizer, oxford])
print(b)
plt.pie(b, labels=a, autopct="%0.1f %%")
plt.gcf().set_size_inches(50, 10)
plt.show()
```

```
['Pfizer/BioNTech' 'Oxford/AstraZeneca' 'Sinovac']
[1318669  828000 1723520]
```



Generar histogramas de vacunas por mes de llega y fabricante.

In [8]:
```
from ipykernel import kernelapp as app
lista1=[]
plt.figure(figsize=[18, 8])
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, date in zip(datos_fabri['vaccine'][1:],datos_fabri['total
    fecha_dt = datetime.strptime(date, '%d/%m/%Y')
    if((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month ==
        con =con+int(j)
        fig1.barh(i,con,color='darkblue')
        fig1.set_title('Enero')

    elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month =
        con1 =con1+int(j)
        fig2.barh(i,con1,color='darkcyan')
        fig2.set_title('Febrero')
    elif((i == a[0] or i == a[1] or i == a[2]) and (fecha_dt.month =
        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 =
        con3 =con3+int(j)
        fig4.barh(i,con3)
        fig4.set_title('Abril')
```



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 [9]:
```
lisx=[]
lisy=[]
datos_vacu.columns =['fecha','dosis_total','primera_dosis', 'segund
```

```
fecha_ini=input("Ingrese fecha-inicio\n")
fecha_fin=input("Ingrese fecha-fin\n")
print("\n")
for d,i, j, k in zip(datos_vacu['fecha'][1:],datos_vacu['dosis_tota
    fecha_dt = datetime.strptime(d, '%d/%m/%Y')
    if(datetime.strptime(fecha_ini,'%d/%m/%Y')<=datetime.strptime(d
        lisx.append(d)
        lisy.append(i)
        print('Vacunados\n',i,"-->",d,'\n')
plt.barh(lisx,lisy)
```

```
Ingrese fecha-inicio
10/01/2019
Ingrese fecha-fin
20/04/2021


Vacunados
 108 --> 22/01/2021

Vacunados
 2982 --> 27/01/2021

Vacunados
 6228 --> 04/02/2021

Vacunados
 8190 --> 17/02/2021

Vacunados
```

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.

Regresion Lineal----------

In [12]:

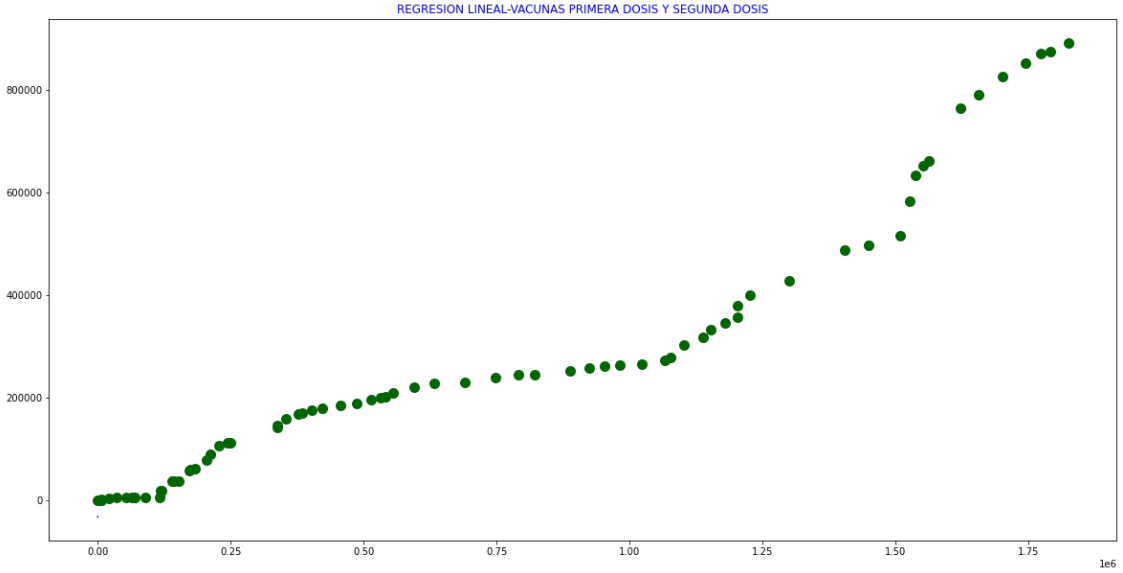
```
#datos_vacu
lista_total_pr = []
lista_total_se = []
for i, j in zip(datos_vacu['primera_dosis'][1:], datos_vacu['segunda
    lista_total_pr.append(i)
    lista_total_se.append(j)
da = np.array(lista_total_pr, dtype='int')

to = np.array(lista_total_se, dtype='int')

# Creamos el objeto de Regresión Lineal
reg = linear_model.LinearRegression()

# Entrenamos nuestro modelo
reg.fit(np.array(da).reshape(-1, 1), to)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tang
print('Coefficients: \n', reg.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', reg.intercept_)
# Error Cuadrado Medio
```

```
Coefficients:
 [0.4146988]
Independent term:
 -31375.929087932513
```

In [14]:
```
y_prediccion = reg.predict([[100]])
print(int(y_prediccion))
```

```
-31334
```

In [18]:
```
plt.scatter(da, to, color='darkgreen', lw=5)
plt.title('REGRESION LINEAL-VACUNAS PRIMERA DOSIS Y SEGUNDA DOSIS',
ree = np.array(range(1, len(da) + 1))
plt.plot(ree, reg.predict(ree.reshape(-1, 1)), color='darkblue')
plt.gcf().set_size_inches(20, 10)
```



In [ ]:
```
lista_total_arr = []
lista_total_to = []
FMT = '%d/%m/%Y'
datev = datos_fabri['arrived_at'][1:]
datos_fabri['arrived_at'] = datev.map(lambda x : (datetime.datetime
for i, j in zip(datos_fabri['arrived_at'][1:], datos_fabri['total']
    lista_total_arr.append(i)
    lista_total_to.append(j)
dav = np.array(lista_total_arr, dtype='int')

tov = np.array(lista_total_to, dtype='int')
```

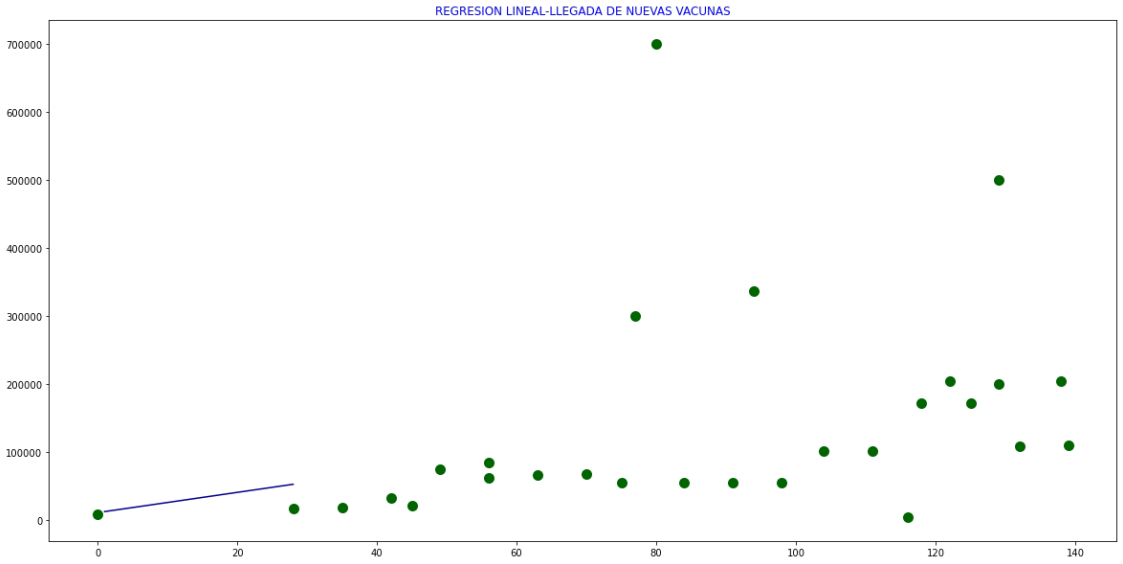
In [22]:
```
# Creamos el objeto de Regresión Lineal
regv = linear_model.LinearRegression()

# Entrenamos nuestro modelo
regv.fit(np.array(dav).reshape(-1, 1), tov)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tan
print('Coefficients: \n', regv.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regv.intercept_)
# Error Cuadrado Medio
```

Coefficients

In [23]:
```
y_prediccionv = regv.predict([[100]])
print(int(y_prediccionv))
```

```
159165
```

In [27]:
```
plt.scatter(dav, tov, color='darkgreen', lw=5)
plt.title('REGRESION LINEAL-LLEGADA DE NUEVAS VACUNAS', color='blue
reev = np.array(range(1, len(dav) + 1))
plt.plot(reev, regv.predict(reev.reshape(-1, 1)), color='darkblue')
plt.gcf().set_size_inches(20, 10)
```

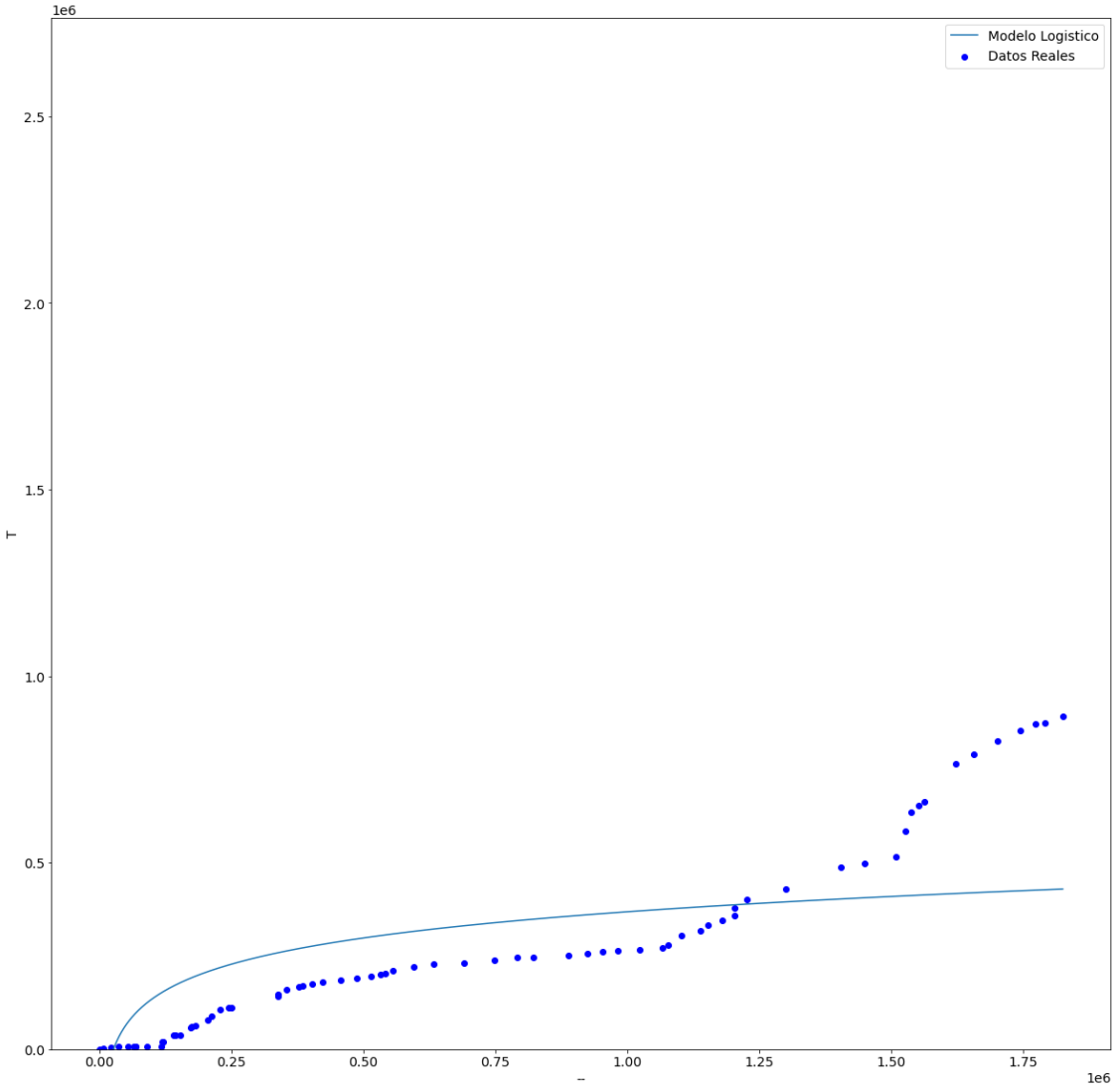


Regresion logaritmica

In [28]:
```
def modelo_logistico(da,a,b):
    return a+b*np.log(da)

exp_fit = curve_fit(modelo_logistico,da,to) #Extraemos los valores
print(exp_fit)
```

```
(array([-1028998.72029681,   101145.62013717]), array([[ 2.6672327
8e+10, -2.05050119e+09],
       [-2.05050119e+09,  1.60436062e+08]]))
```

In [29]:
```
pred_x = list(range(min(da.astype(int)),max(da.astype(int))+50)) # I
plt.rcParams['figure.figsize'] = [7, 7]
plt.rc('font', size=14)
# Real data
plt.scatter(da,to,label="Datos Reales",color="blue")
# Predicted exponential curve
plt.plot(pred_x, [modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) fo
plt.legend()
plt.gcf().set_size_inches(20, 20)
plt.xlabel("--")
plt.ylabel("T")
plt.ylim((min(to)*0.9,max(to)*3.1)) # Definir los limites de Y
plt.show()
```



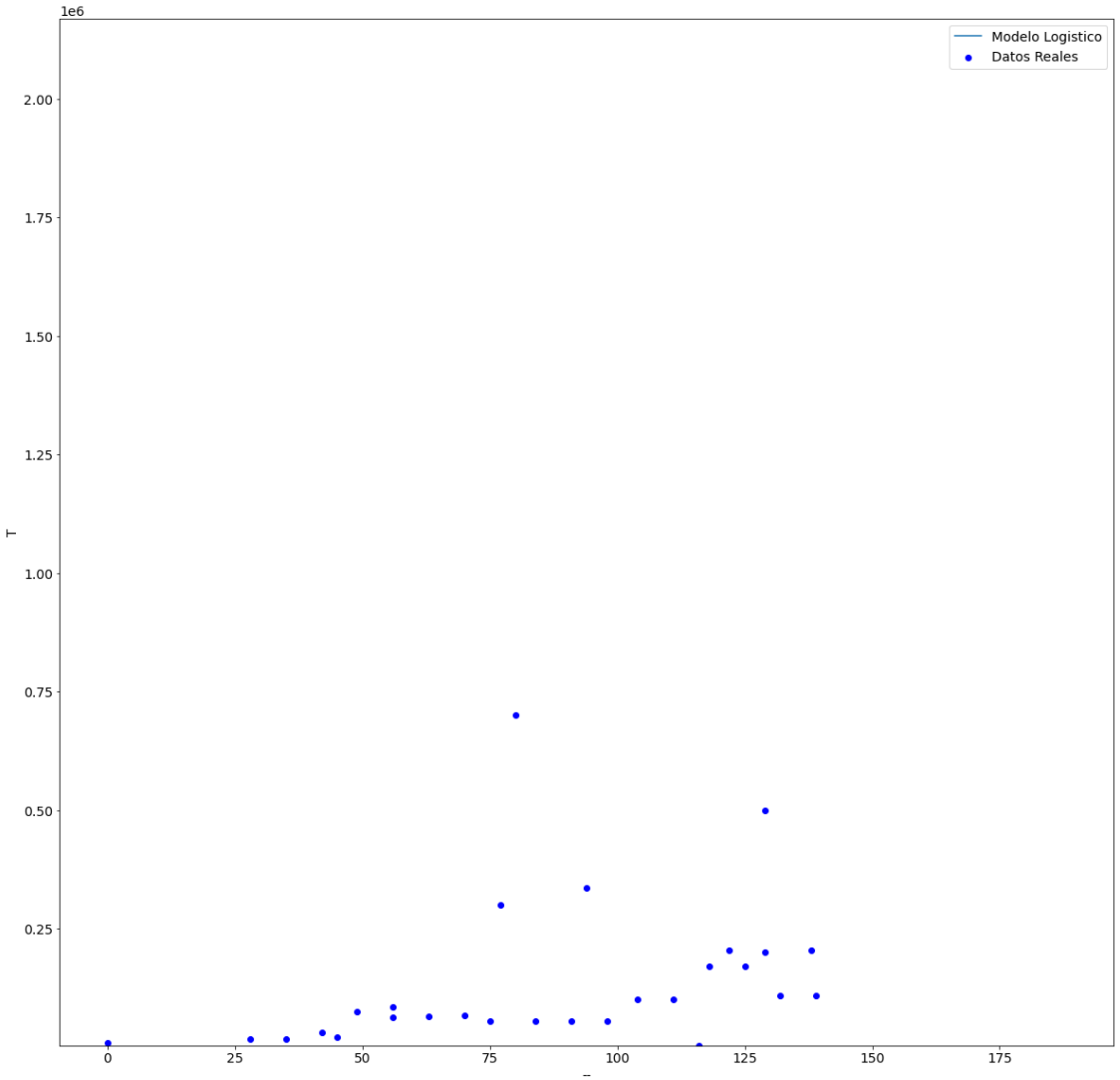
In [30]:
```
def modelo_logisticov(dav,a,b):
    return a+b*np.log(dav)

exp_fitv = curve_fit(modelo_logisticov,dav,tov) #Extraemos los valo
print(exp_fitv)
```

```
(array([1., 1.]), array([[inf, inf],
       [inf, inf]]))
```

```
<ipython-input-30-be02d95d58b3>:2: RuntimeWarning: divide by zero
encountered in log
  return a+b*np.log(dav)
```

In [32]:
```
pred_xv = list(range(min(dav.astype(int)),max(dav.astype(int))+50))
plt.rcParams['figure.figsize'] = [7, 7]
plt.rc('font', size=14)
# Real data
plt.scatter(dav,tov,label="Datos Reales",color="blue")
# Predicted exponential curve
plt.plot(pred_xv, [modelo_logisticov(i,exp_fitv[0][0],exp_fitv[0][1
plt.legend()
plt.gcf().set_size_inches(20, 20)
plt.xlabel("--")
plt.ylabel("T")
plt.ylim((min(tov)*0.9,max(tov)*3.1)) # Definir los limites de Y
plt.show()
```

```
<ipython-input-30-be02d95d58b3>:2: RuntimeWarning: divide by zero
encountered in log
  return a+b*np.log(dav)
```

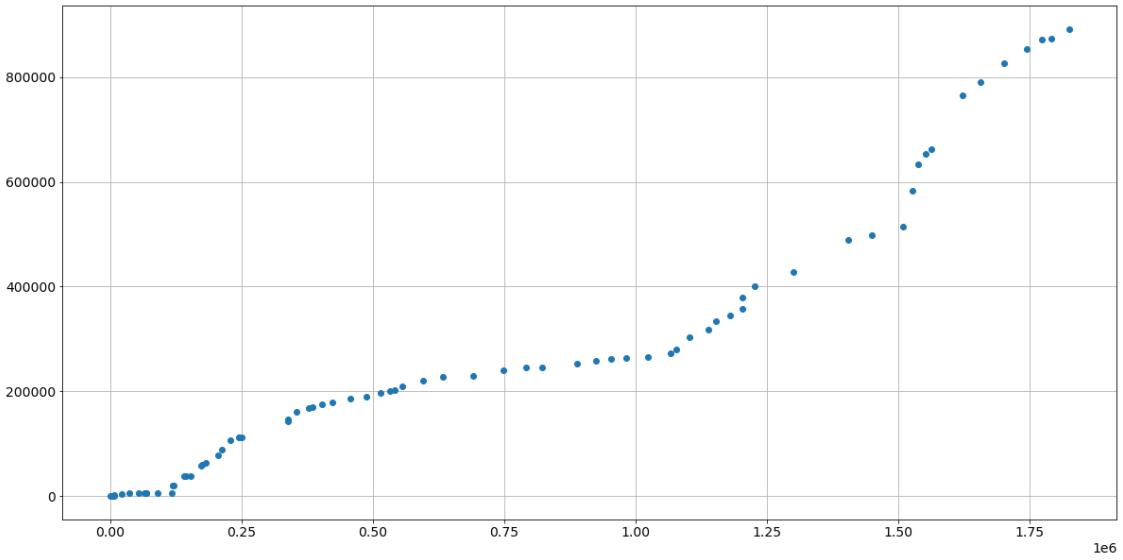


Regrecion Exponencial

In [34]:
```
# Implementar
curve_fit = np.polyfit(da, np.log(to), deg=1)
```

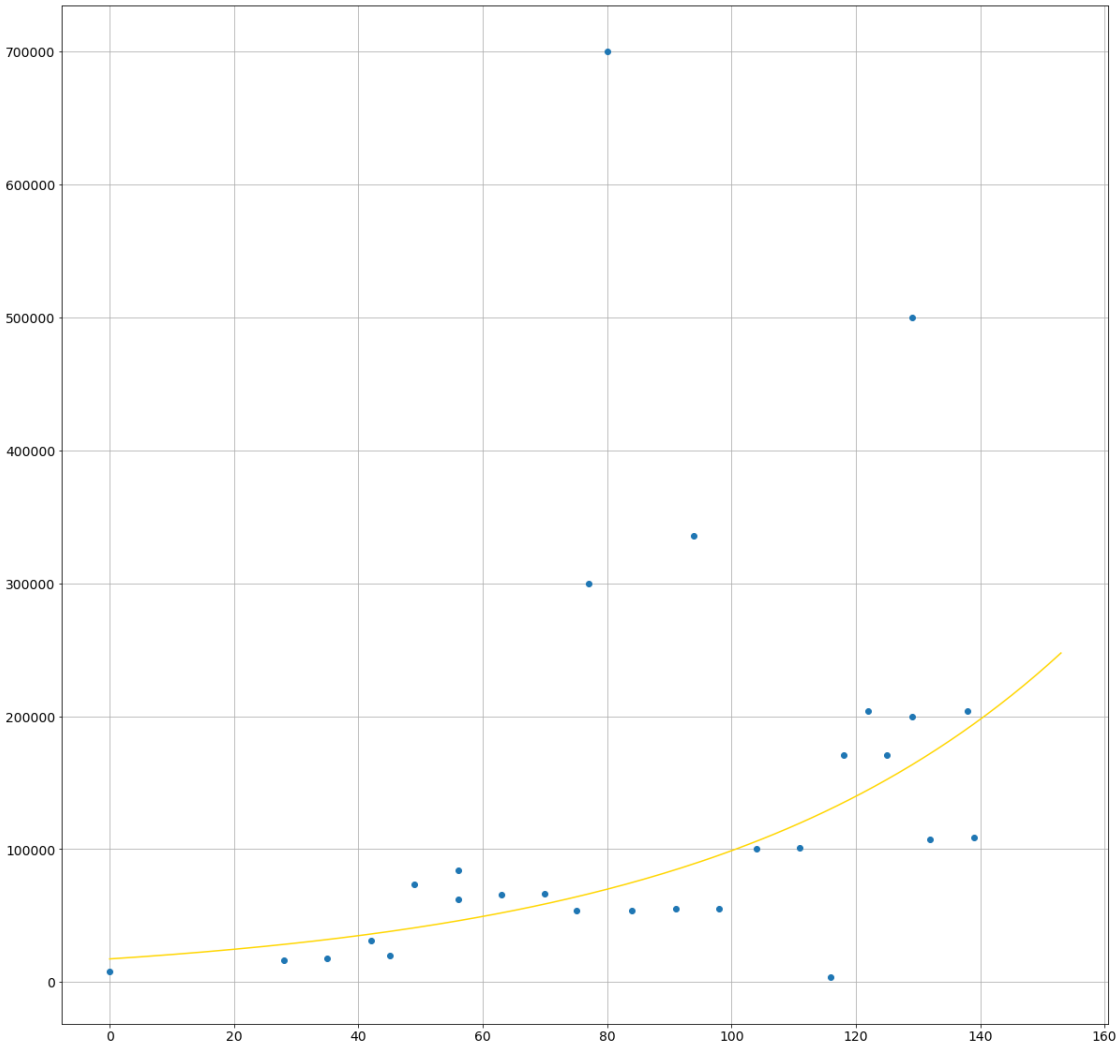
```
print(curve_fit)
pred_x = np.array(list(range(min(da), max(da)+15)))
yx = np.exp(curve_fit[1]) * np.exp(curve_fit[0]*pred_x)
plt.plot(da,to,"o")
plt.plot(pred_x,yx, color="blue")
plt.gcf().set_size_inches(20, 10)
plt.grid(True)
```

```
<ipython-input-34-147c24544348>:2: RuntimeWarning: divide by zero
encountered in log
  curve_fit = np.polyfit(da, np.log(to), deg=1)

[nan nan]
```



In [35]:
```
# Implementar
curve_fitv = np.polyfit(dav, np.log(tov), deg=1)
print(curve_fitv)
pred_xv = np.array(list(range(min(dav), max(dav)+15)))
yxv = np.exp(curve_fitv[1]) * np.exp(curve_fitv[0]*pred_xv)
plt.plot(dav,tov,"o")
plt.plot(pred_xv,yxv, color="gold")
plt.gcf().set_size_inches(20, 20)
plt.grid(True)
```

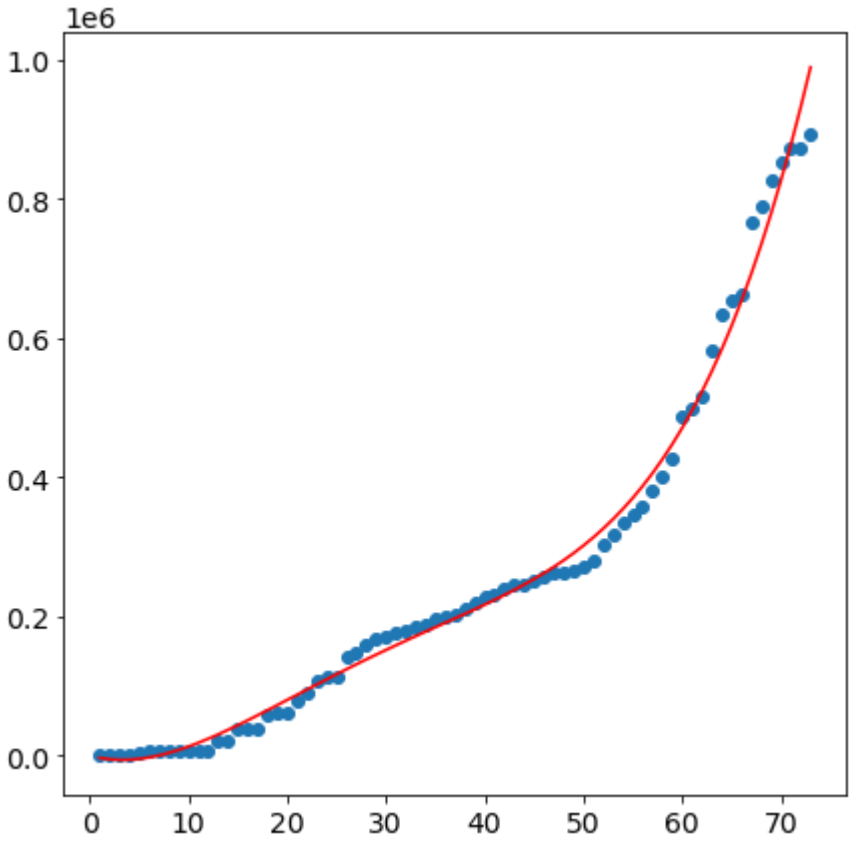
```
[0.01732365 9.76885689]
```



Regrecion Polinomial

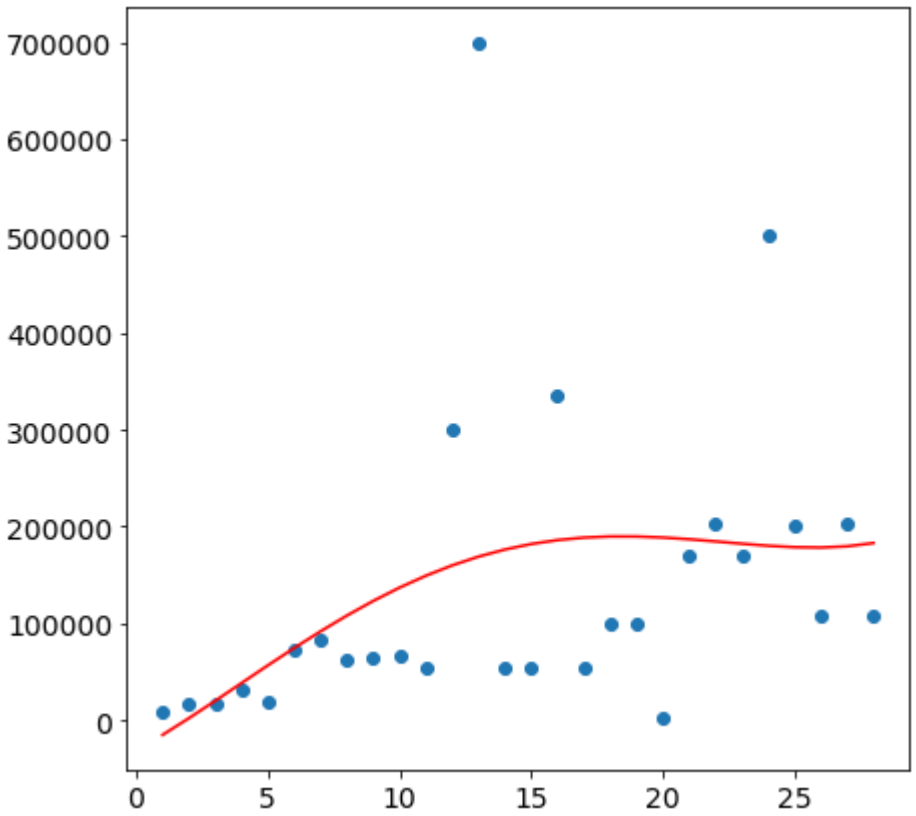
In [36]:
```
xpol=np.arange(1, len(da)+1, 1)

fun1 = np.poly1d(np.polyfit(xpol, to, 4))
print(fun1)
y_pred=fun1(xpol)
plt.scatter(xpol, to)
plt.plot(xpol, y_pred, c='r')
plt.show()
```

```
        4         3         2
0.158 x - 17.71 x + 691.8 x - 4024 x + 193.4
```



In [37]:
```
xpolv=np.arange(1, len(dav)+1, 1)

fun1v = np.poly1d(np.polyfit(xpolv, tov, 4))
print(fun1v)
y_predv=fun1v(xpolv)
plt.scatter(xpolv, tov)
plt.plot(xpolv, y_predv, c='r')
plt.show()
```

```
       4         3        2
1.645 x - 86.3 x + 856 x + 1.523e+04 x - 3.105e+04
```



In [ ]:

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

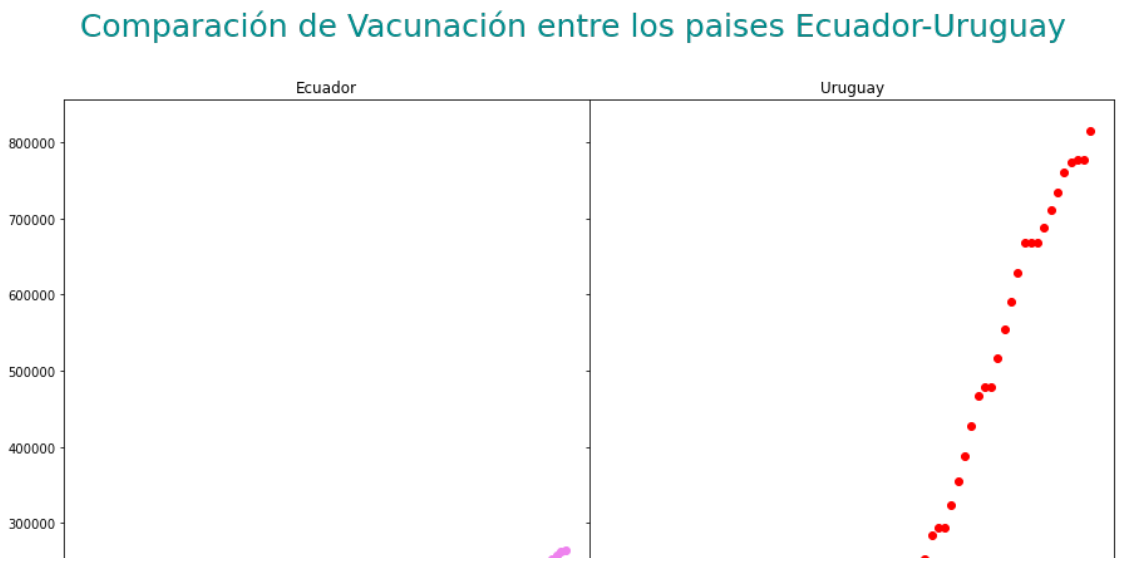
Comparacion de Ecuador y Uruguay

In [20]:
```
datos = 'https://raw.githubusercontent.com/owid/covid-19-data/maste
df_vacum = pd.read_csv(datos, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinat
                    'daily_vaccinations_raw', 'daily_vaccinations',
                    'people_vaccinated_per_hundred', 'people_fully_
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum
    if i == 'Ecuador':
```

```
            lista_fechas.append(j)
            lista_tot.append(k)
        elif i == 'Uruguay':
            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("---------------Uruguay---------------")
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_)
#Grafica
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                               gridspec_kw={'hspace': 0, 'wspace': 0}, figs
fig.suptitle('Comparación de Vacunación entre los paises Ecuador-Uru
ax1.scatter(x, tot, color='violet')
ax1.set_title('Ecuador')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('Uruguay')
```

```
-----------------Ecuador--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
---------------Uruguay---------------
Coefficients:
 [11797.4404233]
Independent term:
 -210623.81621004568
```

Out[20]: Text(0.5, 1.0, 'Uruguay')

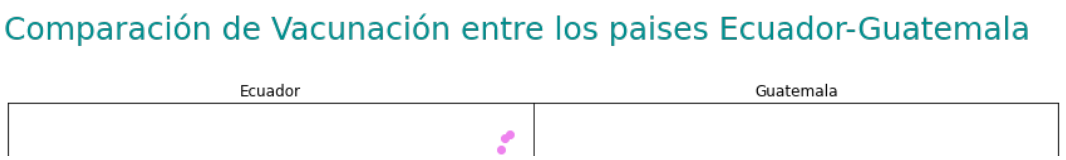


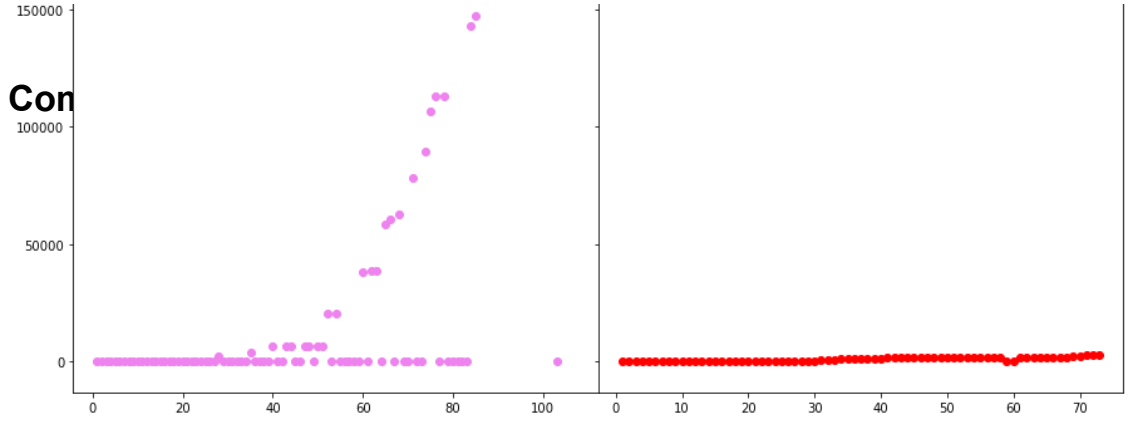
Comparacion entre Ecuador y Guatemala

In [23]:
```
datos = 'https://raw.githubusercontent.com/owid/covid-19-data/maste
df_vacum = pd.read_csv(datos, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinat
                    'daily_vaccinations_raw', 'daily_vaccinations',
                    'people_vaccinated_per_hundred', 'people_fully_
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum
    if i == 'Ecuador':
        lista_fechas.append(j)
        lista_tot.append(k)
    elif i == 'Guatemala':
        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("----------------Guatemala---------------")
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_)
#Grafica
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                          gridspec_kw={'hspace': 0, 'wspace': 0}, figs
fig.suptitle('Comparación de Vacunación entre los paises Ecuador-Gua
ax1.scatter(x, tot, color='violet')
ax1.set_title('Ecuador')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('Guatemala')
```

```
-----------------Ecuador--------------
Coefficients:
 [2111.08278963]
Independent term:
 -61119.23699523892
----------------Guatemala---------------
Coefficients:
 [37.35977416]
Independent term:
 -453.5993150684932
```

Out[23]: Text(0.5, 1.0, 'Guatemala')

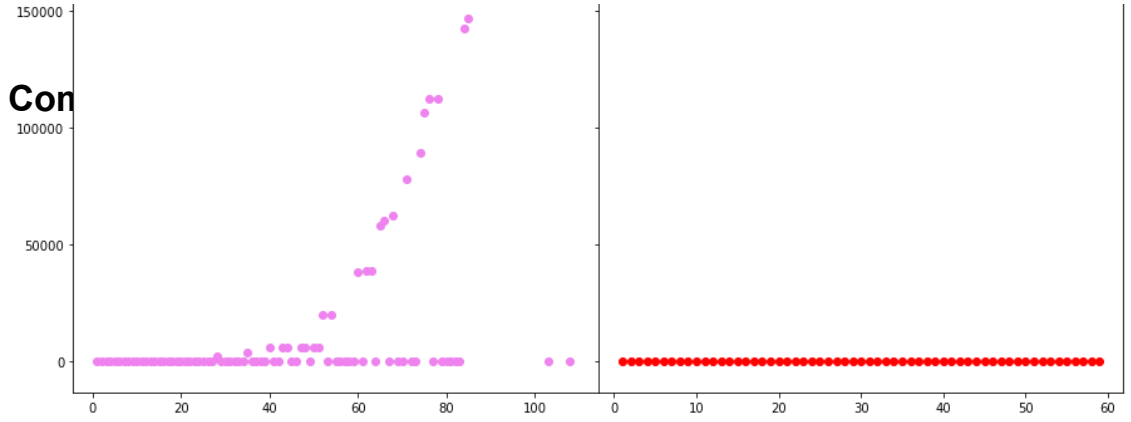



Con
150000
100000
50000
0
0
20
40
60
80
100
0
10
20
30
40
50
60
70

In [30]:
```
datos = 'https://raw.githubusercontent.com/owid/covid-19-data/maste
df_vacum = pd.read_csv(datos, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinat
                    'daily_vaccinations_raw', 'daily_vaccinations',
                    'people_vaccinated_per_hundred', 'people_fully_
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum
    if i == 'Ecuador':
        lista_fechas.append(j)
        lista_tot.append(k)
    elif i == 'Georgia':
        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("---------------Georgia---------------")
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_)
#Grafica
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                          gridspec_kw={'hspace': 0, 'wspace': 0}, figs
fig.suptitle('Comparación de Vacunación entre los paises Ecuador-Ge
ax1.scatter(x, tot, color='violet')
ax1.set_title('Ecuador')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('Georgia')
```

```
-----------------Ecuador--------------
Coefficients:
 [2077.58729497]
Independent term:
 -59919.42048929665
---------------Georgia---------------
Coefficients:
 [0.]
Independent term:
 0.0
```

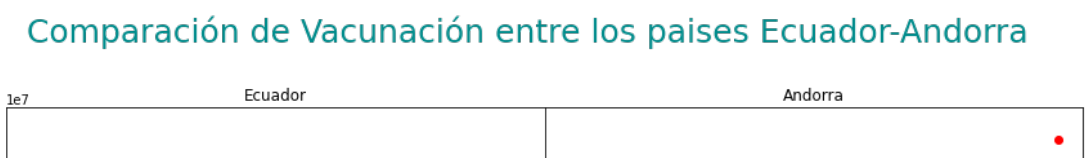
Out[30]: Text(0.5, 1.0, 'Georgia')


Con
150000
100000
50000
0
0
20
40
60
80
100
0
10
20
30
40
50
60

In [32]:
```
datos = 'https://raw.githubusercontent.com/owid/covid-19-data/maste
df_vacum = pd.read_csv(datos, header = None).fillna(0)
df_vacum.columns = ['location', 'iso_code', 'date', 'total_vaccinat
                    'daily_vaccinations_raw', 'daily_vaccinations',
                    'people_vaccinated_per_hundred', 'people_fully_
lista_fechas = []
lista_tot = []
lista_fechasbr = []
lista_totbr = []
for i, j, k in zip(df_vacum['location'], df_vacum['date'], df_vacum
    if i == 'Ecuador':
        lista_fechas.append(j)
        lista_tot.append(k)
    elif i == 'Andorra':
        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("----------------Andorra---------------")
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_)
#Grafica
fig, (ax1, ax2) = plt.subplots(1, 2, sharex='col', sharey='row',
                          gridspec_kw={'hspace': 0, 'wspace': 0}, figs
fig.suptitle('Comparación de Vacunación entre los paises Ecuador-And
ax1.scatter(x, tot, color='violet')
ax1.set_title('Ecuador')
x_real = np.array(range(50, 100))
ax2.scatter(xbr, tot_br, color='red')
ax2.set_title('Andorra')
```

```
-----------------Ecuador--------------
Coefficients:
 [2077.58729497]
Independent term:
 -59919.42048929665
----------------Andorra---------------
Coefficients:
 [254920.07295928]
Independent term:
 -6639228.278956678
```

Out[32]: Text(0.5, 1.0, 'Andorra')



In [ ]: