Prueba 1

In [34]:
```
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
from xml.dom import minidom
```

In [30]:
```
url = 'https://covid.ourworldindata.org/data/owid-covid-data.csv'
df = pd.read_csv(url)
df= df.fillna(1)
df
```

23	AFG	Asia	Afghanistan	2020-01-23	1.0	0.0	0.0	1.0
24	AFG	Asia	Afghanistan	2020-01-24	1.0	0.0	0.0	1.0
25	AFG	Asia	Afghanistan	2020-01-25	1.0	0.0	0.0	1.0
26	AFG	Asia	Afghanistan	2020-01-26	1.0	0.0	0.0	1.0
27	AFG	Asia	Afghanistan	2020-01-27	1.0	0.0	0.0	1.0
28	AFG	Asia	Afghanistan	2020-01-28	1.0	0.0	0.0	1.0
29	AFG	Asia	Afghanistan	2020-01-29	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...

In [31]:
```
df = df[df['location'].isin(['Nicaragua'])]
df = df.loc[:,['date','total_cases','iso_code']]
FMT = '%Y-%m-%d'
date = df['date']
df['date'] = date.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2019-12-3(
df
```

Out[31]:

	date	total_cases	iso_code
38578	80	1.0	NIC
38579	81	1.0	NIC
38580	82	2.0	NIC
38581	83	2.0	NIC
38582	84	2.0	NIC
38583	85	2.0	NIC
38584	86	2.0	NIC
38585	87	2.0	NIC
38586	88	2.0	NIC
38587	89	2.0	NIC
38588	90	4.0	NIC
38589	91	4.0	NIC
38590	92	4.0	NIC
38591	93	4.0	NIC
38592	94	5.0	NIC
38593	95	5.0	NIC
38594	96	5.0	NIC
38595	97	5.0	NIC
38596	98	5.0	NIC
38597	99	6.0	NIC
38598	100	6.0	NIC
38599	101	6.0	NIC
38600	102	6.0	NIC
38601	103	7.0	NIC
38602	104	7.0	NIC
38603	105	9.0	NIC
38604	106	9.0	NIC
38605	107	9.0	NIC
38606	108	9.0	NIC
38607	109	9.0	NIC
...	...	...	...
38799	301	5434.0	NIC
38800	302	5434.0	NIC
38801	303	5514.0	NIC
38802	304	5514.0	NIC
38803	305	5514.0	NIC
38804	306	5514.0	NIC
38805	307	5514.0	NIC
38806	308	5514.0	NIC
38807	309	5514.0	NIC
38808	310	5514.0	NIC
38809	311	5514.0	NIC
38810	312	5591.0	NIC
38811	313	5591.0	NIC
38812	314	5591.0	NIC

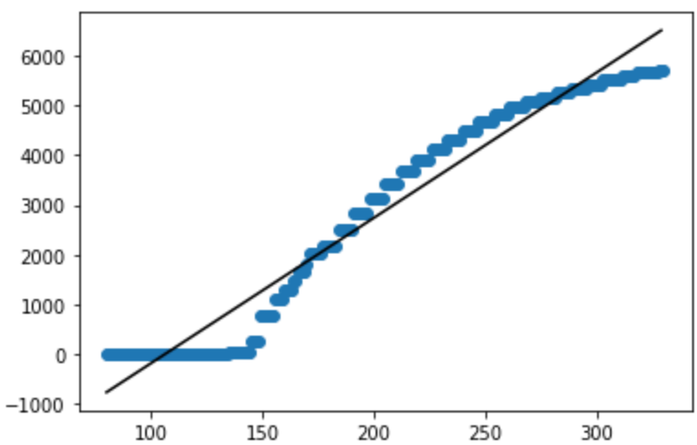
	date	total_cases	iso_code
38813	315	5591.0	NIC
38814	316	5591.0	NIC
38815	317	5591.0	NIC
38816	318	5661.0	NIC
38817	319	5661.0	NIC
38818	320	5661.0	NIC
38819	321	5661.0	NIC
38820	322	5661.0	NIC
38821	323	5661.0	NIC
38822	324	5661.0	NIC
38823	325	5661.0	NIC
38824	326	5661.0	NIC
38825	327	5661.0	NIC
38826	328	5725.0	NIC
38827	329	5725.0	NIC
38828	330	5725.0	NIC

251 rows × 3 columns

Regresion Lineal

In [16]:
```
x = list(df.iloc [:, 0])
y = list(df.iloc [:, 1])
regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1) ,y)
print("Ecuación")
print('m = ' + str(regr.coef_) + ' b = ' + str(regr.intercept_))
print("Predicción")
y_predict = regr.predict([[len(x)+7]])
print(y_predict)
plt.scatter(x,y)
x_real = np.array(range(min(x),max(x)))
plt.plot(x_real, regr.predict(x_real.reshape(-1,1)), color='black')
plt.show()
```

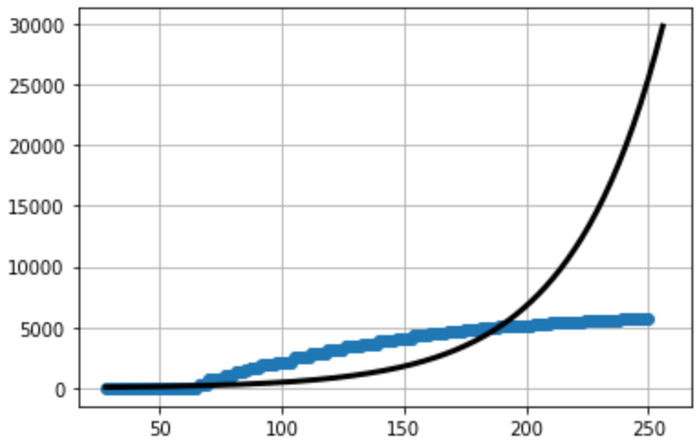
```
Ecuación
m = [29.22685411] b = -3104.023020299753
Predicción
[4436.5053394]
```



Regresion exponencial

In [32]:
```
xexp = np.arange(1,len(df)+1,1)
yexp = np.array(df.values[:,1],dtype='float')
xexp = xexp[27:len(xexp)-1]
yexp = yexp[27:len(yexp)-1]
curve_fit=np.polyfit(xexp,np.log(yexp),1)
pred_x=np.array(list(range(min(xexp),max(xexp)+7)))
yx=np.exp(curve_fit[1])*np.exp(curve_fit[0]*pred_x)
print("Predicción")
print(yx[len(yx)-1])
plt.plot(xexp,yexp,"o")
plt.plot(pred_x,yx,color='black',linewidth=3.0)
plt.grid(True)
plt.show()
```

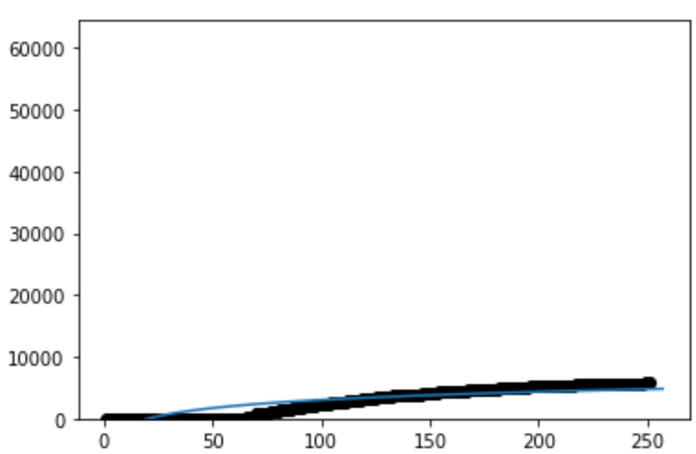
```
Predicción
29806.712715412792
```



Regresion Polinomico

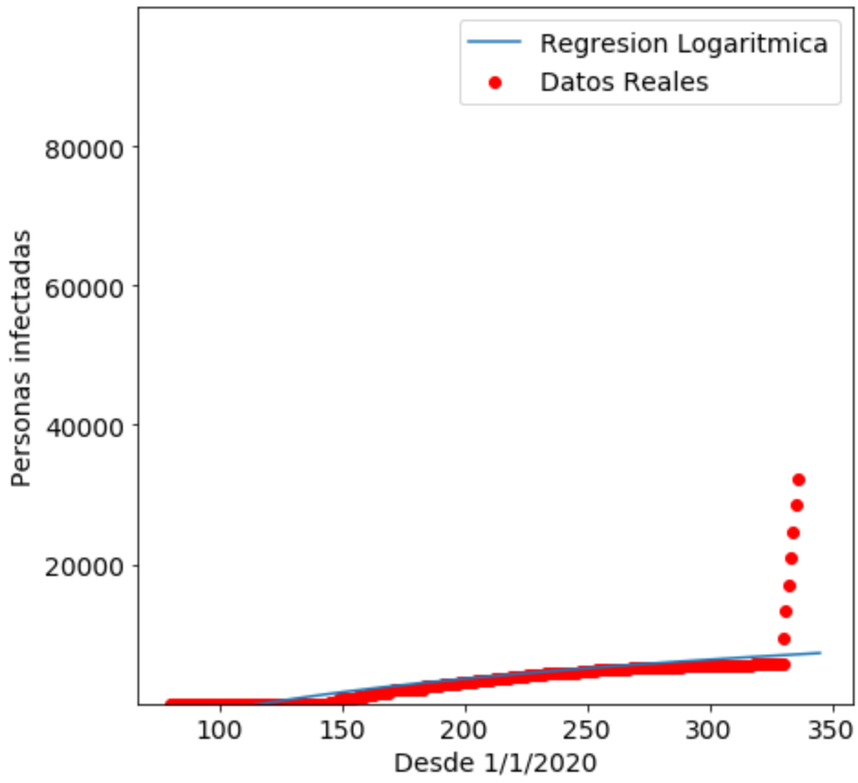
In [35]:
```
def modelo_logistico(x,a,b):
    return a+b*np.log(x)
xlog = np.arange(1,len(df)+1,1)
ylog = np.array(df.values[:,1])
exp_fit = curve_fit(modelo_logistico,xlog,ylog)
pred_x = list(range(min(xlog),max(xlog)+7))
plt.scatter(xlog,ylog,color='black')
val = [modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) for i in pred_x]
print("Predicción")
print(val[len(pred_x)-1])
plt.plot(pred_x, [modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) for i in pred_x])
plt.ylim(0,max(y)*2)
plt.show()
```

```
Predicción
4812.759467256569
```



Regresion Logaritmica

In [40]:
```
def modelo_logistico(x,a,b):
    res= a+b*np.log(x)
    return res
exp_fit = curve_fit(modelo_logistico,x,y)
pred_x = list(range(min(x),max(x)+10))
plt.rcParams['figure.figsize'] = [7, 7]
plt.rc('font', size=14)
plt.scatter(x,y,label="Datos Reales",color="red")
plt.plot(pred_x, [modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) for i in pred_x], label='
plt.legend()
plt.xlabel("Desde 1/1/2020")
plt.ylabel("Personas infectadas")
plt.ylim((min(y)*0.9,max(y)*3.1))
plt.show()
print("Contagios proximos 7 dias: ",[modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) for i
```

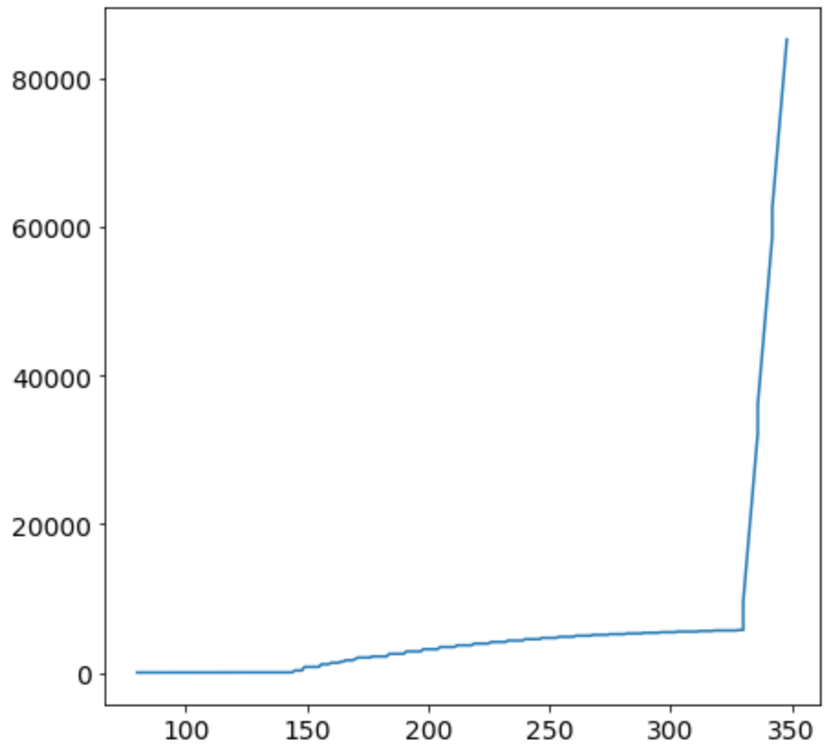



```
Contagios proximos 7 dias:  [7268.221124911819]
```

Modelo con datos

In [42]:
```
filtro = df['total_cases'][27:]
media = filtro.mean()
mediana = filtro.median()
print("media")
print(media)
print("mediana")
print(mediana)
print("Predicción")
print(int(y[-1] + mediana))
for i in range(x[-1], x[-1]+7):
    x.append(i)
    y.append(int(y[-1]+mediana))
predict = int(y[-1] + mediana)
plt.plot(x,y)
plt.show()
print("Predicción")
print(predict)
```

```
media
3235.0133928571427
mediana
3787.0
Predicción
62530
```




```
Predicción
89039
```

Mejor modelo

En base a lo observado el modelo que tiene mejor prediccion es el "Polinomico" ya que este da un resultado mas exacto.

Ventajas y desventajas de los modelos.

-Ventajas
El modelo lineal es facil de implementar
El modelo logaritmico es facil de modelar
EL modelo exponencial no necesita muchos de datos historicos
El modelo polinomial se usa con datos no lienales
-Desventajas
El modelo lineal no se puede utilizar con ecuaciones complejas

El modelo logoritmico solo usa datos lineales
El modelo exponencial no modela correlaciones
El modelo polinomial necesita datos especificos

Cual es el principal problema del modelo probabilistico

Su principal problema es manejar una cantidad de numeros muy grandes, ya que su resultado es inexacto