In [1]:
```python
#Task:1)To draw graph,calculate the r2 score and to predict the daily cases in the month of October
#Program By:Ayush Pandey
#Email Id:1805290@kiit.ac.in
#DATE:18-Oct-2021
#Python Version:3.7
#CAVEATS:None
#LICENSE:None
```

In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```python
#Reading the csv file
df=pd.read_csv('case_time_series.csv')
```

In [4]:
```python
df.head()
```

Out[4]:

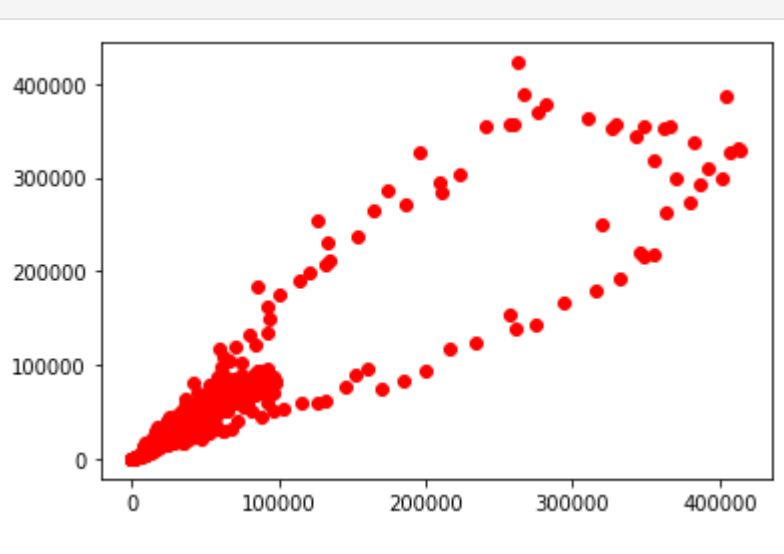
	Date	Date_YMD	Daily Confirmed	Total Confirmed	Daily Recovered	Total Recovered	Daily Deceased	Total Deceased
0	30 January 2020	2020-01-30	1	1	0	0	0	0
1	31 January 2020	2020-01-31	0	1	0	0	0	0
2	1 February 2020	2020-02-01	0	1	0	0	0	0
3	2 February 2020	2020-02-02	1	2	0	0	0	0
4	3 February 2020	2020-02-03	1	3	0	0	0	0

In [ ]:

In [5]:
```python
#Checking whether NaN values are present or not
df.isna().sum()
```

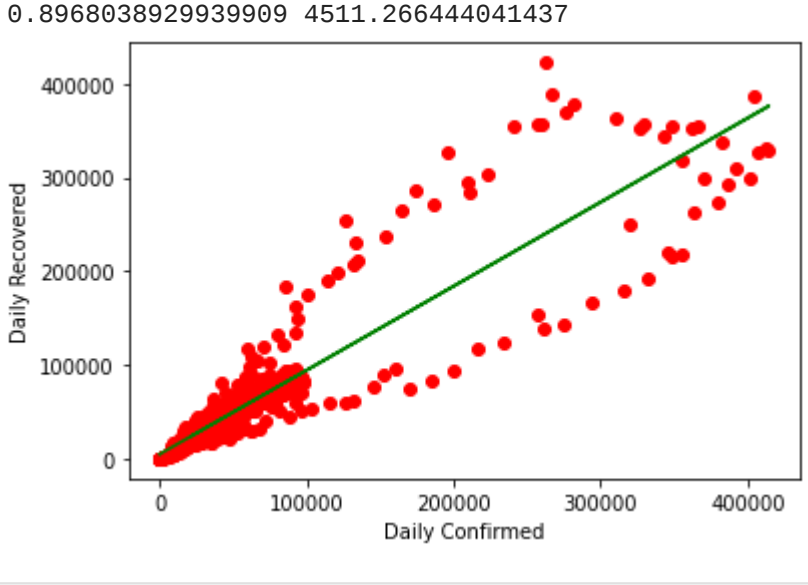
Out[5]:
```
Date               0
Date_YMD           0
Daily Confirmed    0
Total Confirmed    0
Daily Recovered    0
Total Recovered    0
Daily Deceased     0
Total Deceased     0
dtype: int64
```

In [6]:
```python
#Daily Confirmed and Daily Recovered Graph
def graph(x,y):
    plt.scatter(x,y,color='red')
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line),color='green')
    plt.show()
a=df['Daily Confirmed']
s=df['Daily Recovered']
graph(a,s)
```



In [7]:
```python
#Checking for a linear regression between the Daily Confirmed and Daily Recovered
def linear(x,y):
    x_mean=np.mean(x)
    y_mean=np.mean(y)
    x_std=np.std(x)
    x_squared=[]
    x_diff=[]
    y_diff=[]
    for i in range(len(x)):
        x_diff.append(x[i]-x_mean)
        x_squared.append(x_diff[i]**2)
    for i in range(len(x)):
        y_diff.append(y[i]-y_mean)
    see=[]
    for i in range(len(x)):
        see.append(x_diff[i]*y_diff[i])
    m=sum(see)/sum(x_squared)
    c=y_mean-m*x_mean
    return(m,c)
def show(x,y,m,c):
    print(m,c)
    plt.scatter(x,y,color="red")
    predicted=[]
    for i in range(0,len(x)):
        predicted.append(m*x[i]+c)
    plt.plot(x,predicted,color="green")
    plt.xlabel("Daily Confirmed")
    plt.ylabel("Daily Recovered")
    y_predicted=np.array(predicted)
    plt.show()
x=np.array(list(df['Daily Confirmed']))
y=np.array(list(df['Daily Recovered']))
m,c=linear(x,y)
predicted=show(x,y,m,c)
```

```
0.8968038929939909 4511.266444041437
```

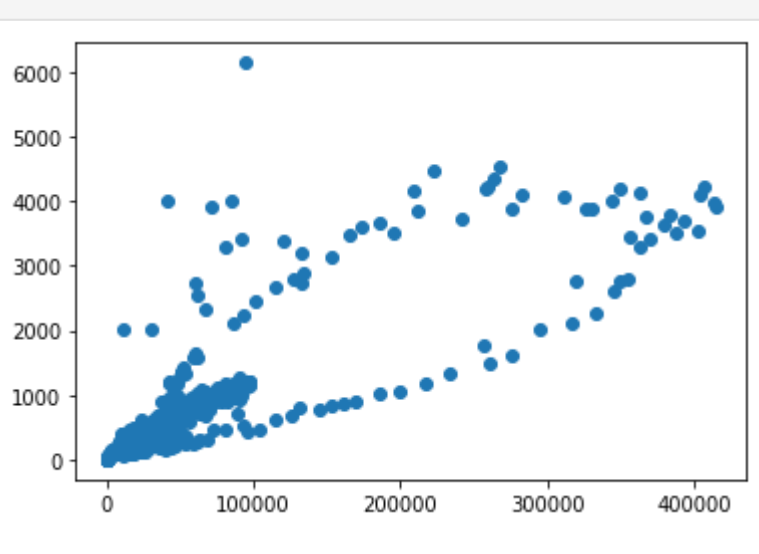


In [8]:
```python
from sklearn.metrics import r2_score
```

In [9]:
```python
#R2 score between Daily Confirmed and Daily Recovered
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

```
0.8615040522307793
```

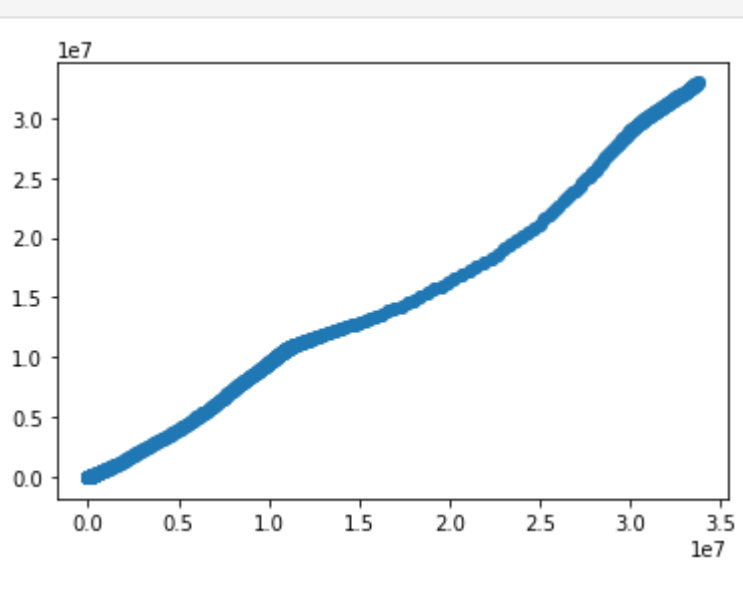
In [10]:
```python
#Daily Confirmed and Daily Deceased Graph
def graph(x,y):
    plt.scatter(x,y)
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line))
    plt.show()
a=df['Daily Confirmed']
s=df['Daily Deceased']
graph(a,s)
```



In [11]:
```python
#R2 score between Daily Confirmed and Daily Deceased
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

```
0.740910399180611
```

In [12]:
```python
#Total Confirmed and Total Recovered Graph
def graph(x,y):
    plt.scatter(x,y)
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line))
    plt.show()
a=df['Total Confirmed']
s=df['Total Recovered']
graph(a,s)
```

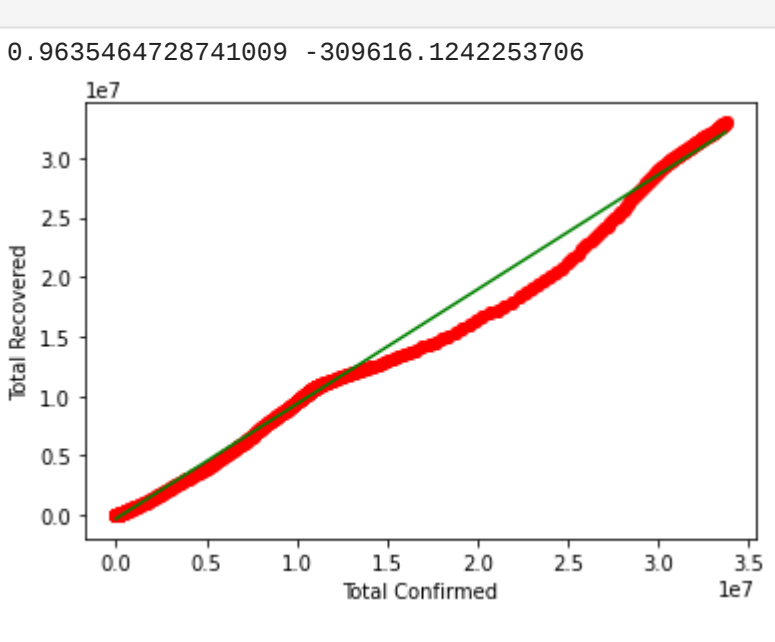


In [13]:
```python
#R2 score between Total Confirmed and Total Recovered
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

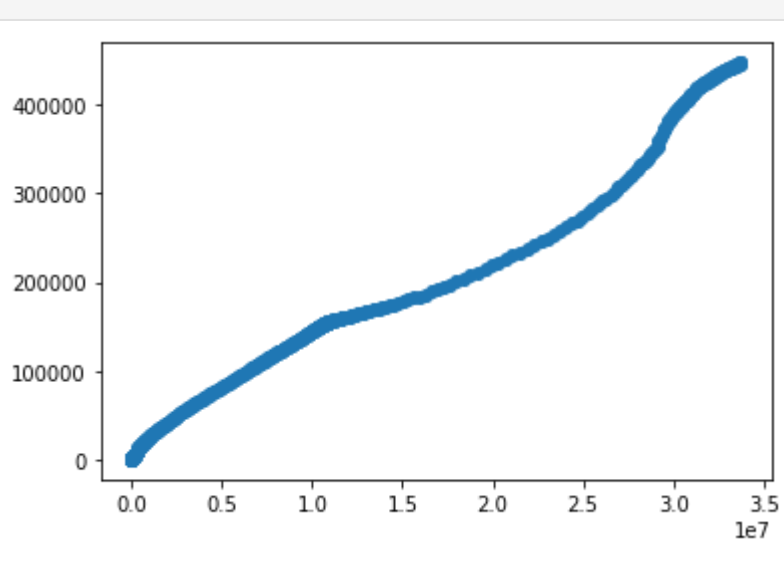
```
0.9981328560121421
```

In [14]:
```python
#Drawing a linear regression between Total Confirmed and Total Recovered
def linear(x,y):
    x_mean=np.mean(x)
    y_mean=np.mean(y)
    x_std=np.std(x)
    x_squared=[]
    x_diff=[]
    y_diff=[]
    for i in range(len(x)):
        x_diff.append(x[i]-x_mean)
        x_squared.append(x_diff[i]**2)
    for i in range(len(x)):
        y_diff.append(y[i]-y_mean)
    see=[]
    for i in range(len(x)):
        see.append(x_diff[i]*y_diff[i])
    m=sum(see)/sum(x_squared)
    c=y_mean-m*x_mean
    return(m,c)
def show(x,y,m,c):
    print(m,c)
    plt.scatter(x,y,color="red")
    predicted=[]
    for i in range(0,len(x)):
        predicted.append(m*x[i]+c)
    plt.plot(x,predicted,color="green")
    plt.xlabel("Total Confirmed")
    plt.ylabel("Total Recovered")
    y_predicted=np.array(predicted)
    plt.show()
x=np.array(list(df['Total Confirmed']))
y=np.array(list(df['Total Recovered']))
m,c=linear(x,y)
predicted=show(x,y,m,c)
```

```
0.9635464728741009 -309616.1242253706
```



In [15]:
```python
#Total Confirmed and Total Deceased Graph
def graph(x,y):
    plt.scatter(x,y)
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line))
    plt.show()
a=df['Total Confirmed']
s=df['Total Deceased']
graph(a,s)
```

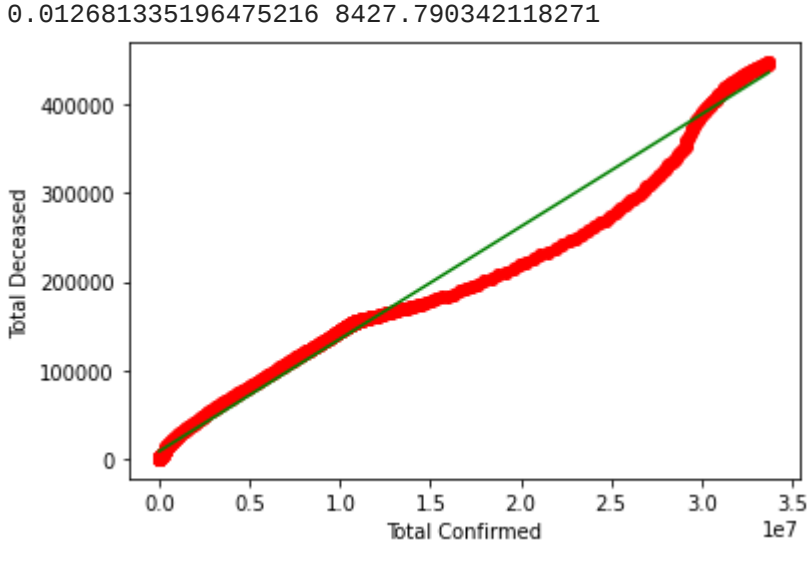


In [16]:
```python
#R2 score between Total Confirmed and Total Deceased
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

```
0.9983348804567995
```

In [17]:
```python
#Drawing a linear regression between Total Confirmed and Total Deceased
def linear(x,y):
    x_mean=np.mean(x)
    y_mean=np.mean(y)
    x_std=np.std(x)
    x_squared=[]
    x_diff=[]
    y_diff=[]
    for i in range(len(x)):
        x_diff.append(x[i]-x_mean)
        x_squared.append(x_diff[i]**2)
    for i in range(len(x)):
        y_diff.append(y[i]-y_mean)
    see=[]
    for i in range(len(x)):
        see.append(x_diff[i]*y_diff[i])
    m=sum(see)/sum(x_squared)
    c=y_mean-m*x_mean
    return(m,c)
def show(x,y,m,c):
    print(m,c)
    plt.scatter(x,y,color="red")
    predicted=[]
    for i in range(0,len(x)):
        predicted.append(m*x[i]+c)
    plt.plot(x,predicted,color="green")
    plt.xlabel("Total Confirmed")
    plt.ylabel("Total Deceased")
    y_predicted=np.array(predicted)
    plt.show()
x=np.array(list(df['Total Confirmed']))
y=np.array(list(df['Total Deceased']))
m,c=linear(x,y)
predicted=show(x,y,m,c)
```

```
0.012681335196475216 8427.790342118271
```



In [18]:
```python
df.tail()
```

Out[18]:

	Date	Date_YMD	Daily Confirmed	Total Confirmed	Daily Recovered	Total Recovered	Daily Deceased	Total Deceased
603	24 September 2021	2021-09-24	29565	33622855	28049	32868599	291	446100
604	25 September 2021	2021-09-25	28169	33651024	26021	32894620	259	446359
605	26 September 2021	2021-09-26	26999	33678023	29625	32924245	276	446635
606	27 September 2021	2021-09-27	14907	33692930	24251	32948496	181	446816
607	28 September 2021	2021-09-28	21898	33714828	29943	32978439	375	447191

In [ ]:

In [19]:
```python
df.dtypes
```

Out[19]:
```
Date               object
Date_YMD           object
Daily Confirmed     int64
Total Confirmed     int64
Daily Recovered     int64
Total Recovered     int64
Daily Deceased      int64
Total Deceased      int64
dtype: object
```

In [20]:
```python
#Extracting the month and year from the Date Column
```

In [21]:
```python
import datetime as dt
df['Date_YMD']=pd.to_datetime(df['Date_YMD'])
df['Month']=df['Date_YMD'].dt.month
df['Year']=df['Date_YMD'].dt.year
```

In [22]:
```python
df.dtypes
```

Out[22]:
```
Date                       object
Date_YMD           datetime64[ns]
Daily Confirmed             int64
Total Confirmed             int64
Daily Recovered             int64
Total Recovered             int64
Daily Deceased              int64
Total Deceased              int64
Month                       int64
Year                        int64
dtype: object
```

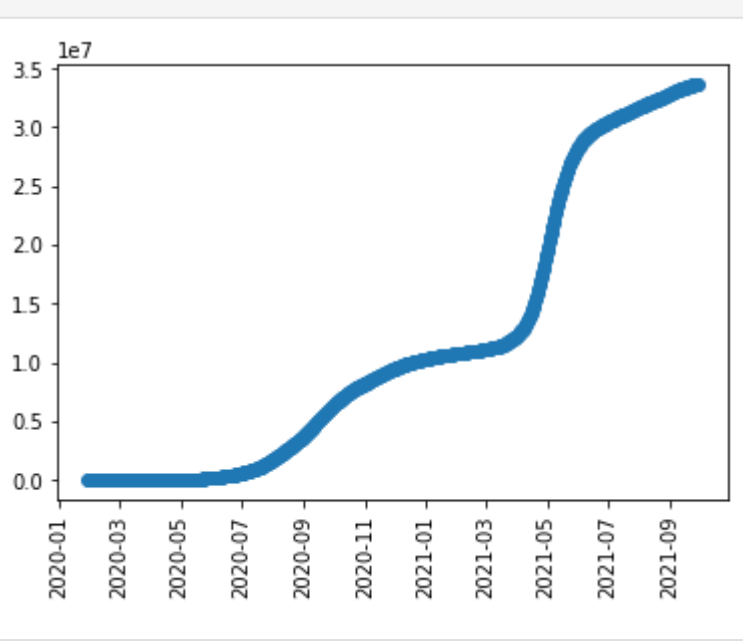
In [23]:
```python
#Grouping based on the Month and Year
df1=df.groupby(['Month','Year'],as_index=False).agg({'Daily Confirmed':'sum','Total Confirmed':'sum','Daily Recovered':'sum','Total Recovered':'su
```

In [24]:
```python
df1.head()
```

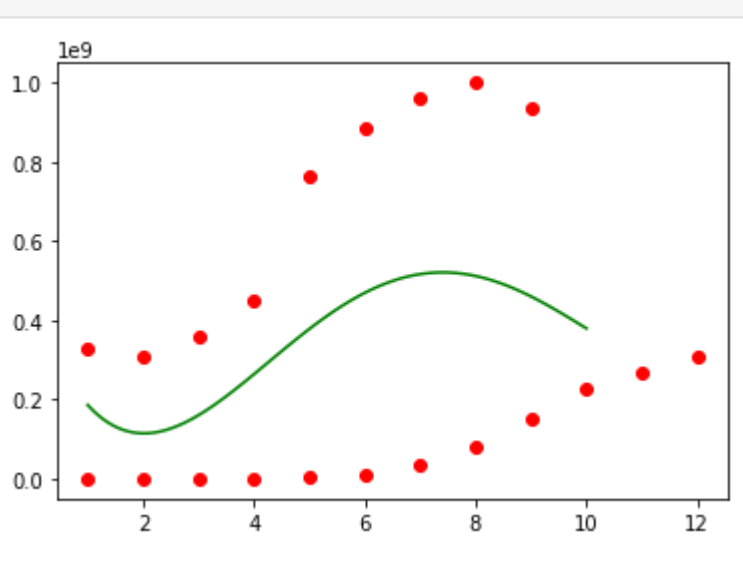
Out[24]:

	Month	Year	Daily Confirmed	Total Confirmed	Daily Recovered	Total Recovered	Daily Deceased	Total Deceased
0	1	2020	1	2	0	0	0	0
1	1	2021	472317	326969299	552275	315855355	5410	4697403
2	2	2020	2	84	3	41	0	0
3	2	2021	353427	305984014	350561	297433362	2766	4346696
4	3	2020	1632	10835	147	912	47	237

In [25]:
```python
plt.scatter(df['Date_YMD'],df['Total Confirmed'])
plt.xticks(rotation=90)
plt.show()
```



In [26]:
```python
#Total Confirmed and Month Graph
def graph(x,y):
    plt.scatter(x,y,color='red')
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line),color='green')
    plt.show()
a=df1['Month']
s=df1['Total Confirmed']
graph(a,s)
```

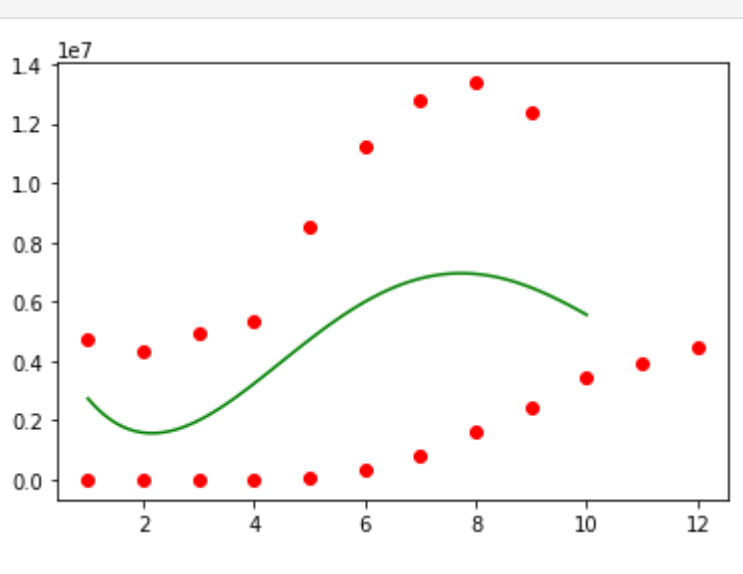


In [27]:
```python
#Prediction of Total Confirmed in the Month of October
def predict(a,s,value):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(poly(value))
print(predict(a,s,10))
```

```
379291387.27002156
```

In [ ]:

In [28]:
```python
#Total Deceased and Month Graph
def graph(x,y):
    plt.scatter(x,y,color='red')
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line),color='green')
    plt.show()
a=df1['Month']
s=df1['Total Deceased']
graph(a,s)
```



In [29]:
```python
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

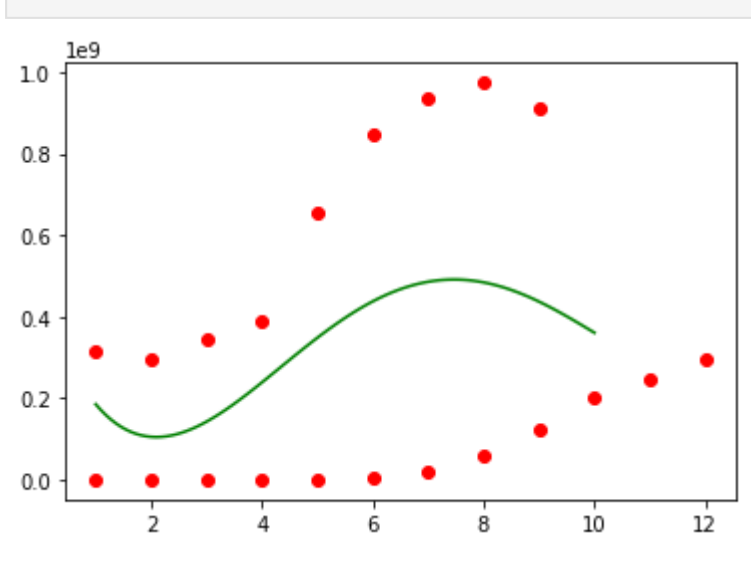
```
0.17998437360156472
```

In [30]:
```python
#Prediction of Total Deceased in the Month of October
def predict(a,s,value):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(poly(value))
print(predict(a,s,10))
```

```
5553160.036567748
```

In [ ]:

In [31]:
```python
#Total Recovered and Month Graph
def graph(x,y):
    plt.scatter(x,y,color='red')
    poly=np.poly1d(np.polyfit(x,y,4))
    line=np.linspace(1,10,100)
    plt.plot(line,poly(line),color='green')
    plt.show()
a=df1['Month']
s=df1['Total Recovered']
graph(a,s)
```



In [32]:
```python
def r_square(a,s):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(r2_score(s,poly(a)))
print(r_square(a,s))
```

```
0.15909067447984404
```

In [33]:
```python
#Prediction of Total Recovered in the Month of October
def predict(a,s,value):
    poly=np.poly1d(np.polyfit(a,s,4))
    return(poly(value))
print(predict(a,s,10))
```

```
360559381.845517
```

In [ ]:

In [ ]: