In [245]:

```
import pandas as pd
```

Extracting the Covid Dataset

In [246]:

```
df=pd.read_csv("covid_dataset.csv")    #loaded the dataset
df.head()
```

Out[246]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_ca
0	AFG	Asia	Afghanistan	2019-12-31	0.0	0.0	0.0	0.0	0.0	
1	AFG	Asia	Afghanistan	2020-01-01	0.0	0.0	0.0	0.0	0.0	
2	AFG	Asia	Afghanistan	2020-01-02	0.0	0.0	0.0	0.0	0.0	
3	AFG	Asia	Afghanistan	2020-01-03	0.0	0.0	0.0	0.0	0.0	
4	AFG	Asia	Afghanistan	2020-01-04	0.0	0.0	0.0	0.0	0.0	

5 rows × 36 columns

In [247]:

```
df=df[df["location"]=="India"]  #Subsetting only those rows which have the df["location"
] as "India"
```

In [248]:

```
df.head()  # testing whether the subsetting has been successful
```

Out[248]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_c
14956	IND	Asia	India	2019-12-31	0.0	0.0	0.0	0.0	0.0	
14957	IND	Asia	India	2020-01-01	0.0	0.0	0.0	0.0	0.0	
14958	IND	Asia	India	2020-01-02	0.0	0.0	0.0	0.0	0.0	
14959	IND	Asia	India	2020-01-03	0.0	0.0	0.0	0.0	0.0	
14960	IND	Asia	India	2020-01-04	0.0	0.0	0.0	0.0	0.0	

5 rows × 36 columns

Handling Missing Values

In [249]:

```
continuous=["new_tests","total_tests_per_thousand","new_tests_per_thousand","new_tests_smoothed_per_thousand","new_tests_smoothed","tests_per_case","positive_rate","stringency_index","total_tests"]
for i in continuous:
    df[i].fillna(value=df[i].mean(),inplace=True)
# for continuous numerical columns we have replaced the null values with the mean of the columns


# there are no ordinal numerical columns in this dataset


categorical=["tests_units"]

for j in categorical:
    df[j].fillna(value="samples tested",inplace=True)
# for categorical columns we have replaced the null values with the mode of that column
```

In [250]:

```
df.info()   # confirming whether there are any null values left
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 222 entries, 14956 to 15177
Data columns (total 36 columns):
iso_code                            222 non-null object
continent                           222 non-null object
location                            222 non-null object
date                                222 non-null object
total_cases                         222 non-null float64
new_cases                           222 non-null float64
total_deaths                        222 non-null float64
new_deaths                          222 non-null float64
total_cases_per_million             222 non-null float64
new_cases_per_million               222 non-null float64
total_deaths_per_million            222 non-null float64
new_deaths_per_million              222 non-null float64
new_tests                           222 non-null float64
total_tests                         222 non-null float64
total_tests_per_thousand            222 non-null float64
new_tests_per_thousand              222 non-null float64
new_tests_smoothed                  222 non-null float64
new_tests_smoothed_per_thousand     222 non-null float64
tests_per_case                      222 non-null float64
positive_rate                       222 non-null float64
tests_units                         222 non-null object
stringency_index                    222 non-null float64
population                          222 non-null float64
population_density                  222 non-null float64
median_age                          222 non-null float64
aged_65_older                       222 non-null float64
aged_70_older                       222 non-null float64
gdp_per_capita                      222 non-null float64
extreme_poverty                     222 non-null float64
cardiovasc_death_rate               222 non-null float64
diabetes_prevalence                 222 non-null float64
female_smokers                      222 non-null float64
male_smokers                        222 non-null float64
handwashing_facilities              222 non-null float64
hospital_beds_per_thousand          222 non-null float64
life_expectancy                     222 non-null float64
dtypes: float64(31), object(5)
memory usage: 64.2+ KB
```

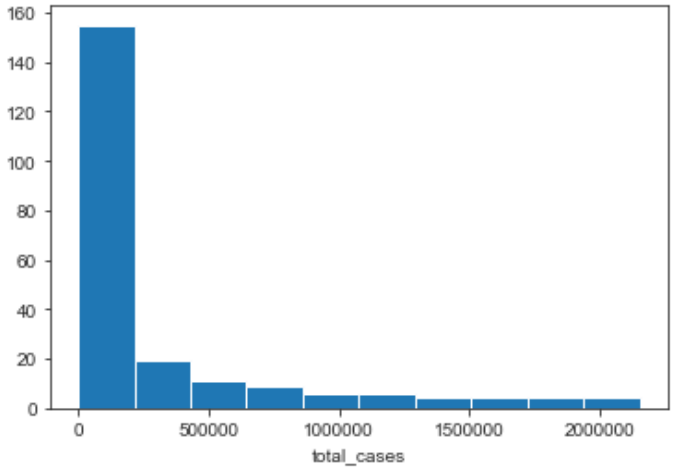
Univariate Analysis

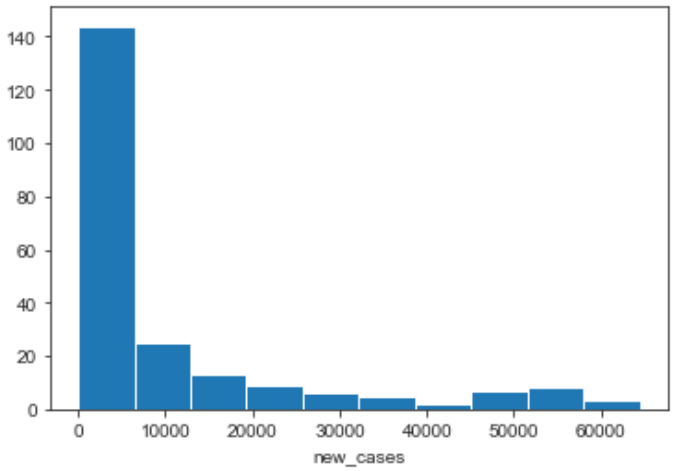
1: Plotting Histograms

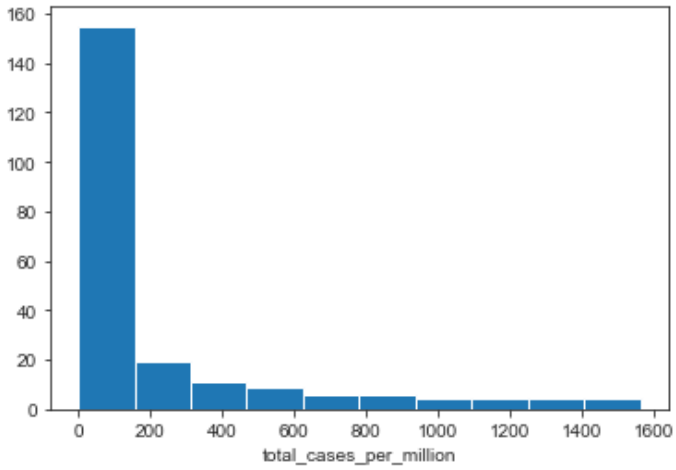
In [251]:

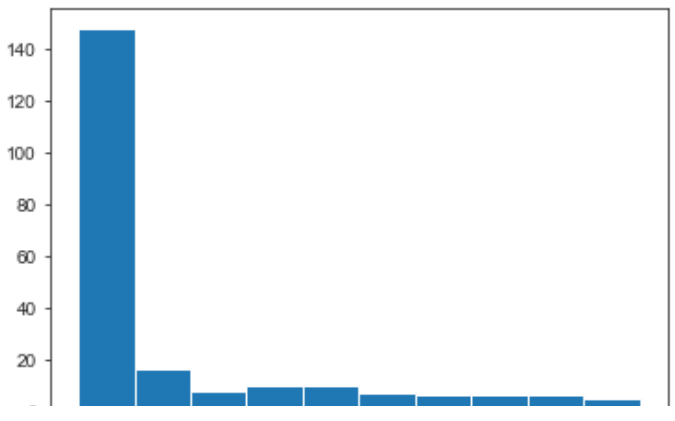
In [251]:

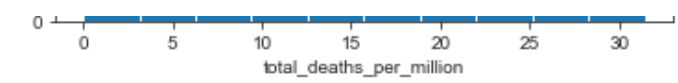
```
import matplotlib.pyplot as plt
hist=['total_cases','new_cases','total_cases_per_million','total_deaths_per_million','new
_tests_smoothed_per_thousand','positive_rate','stringency_index','diabetes_prevalence','l
ife_expectancy','hospital_beds_per_thousand']
for graph in hist:
    plt.hist(df[graph])
    plt.xlabel(graph)
    plt.show()
```

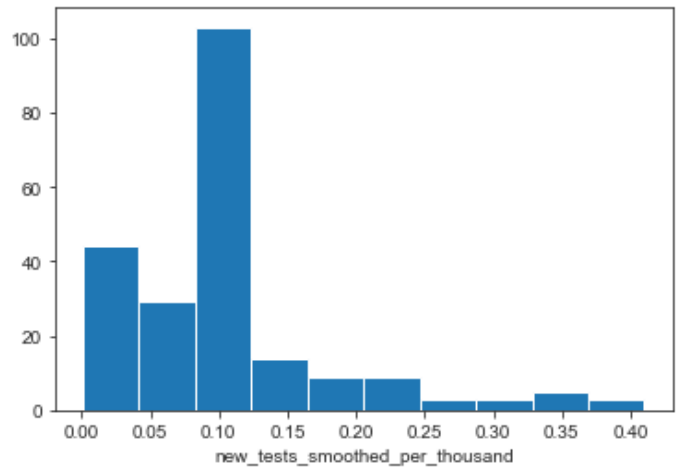


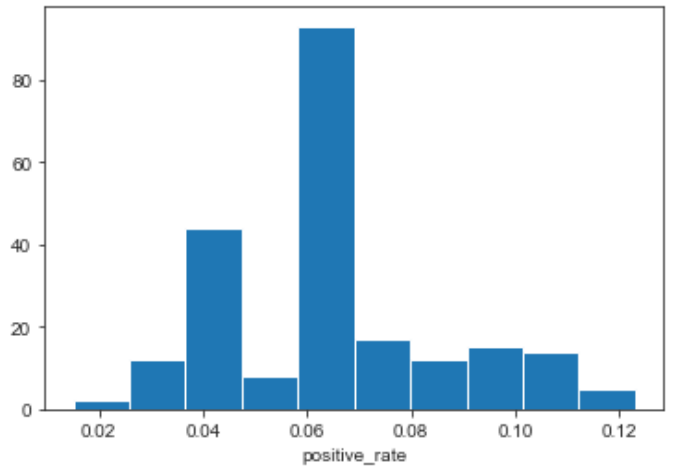


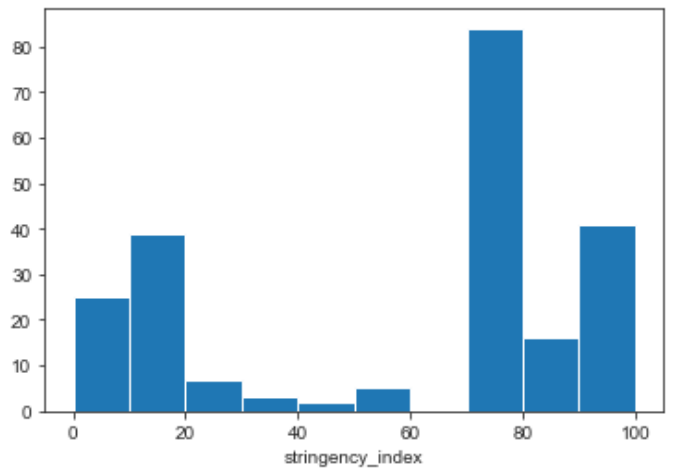


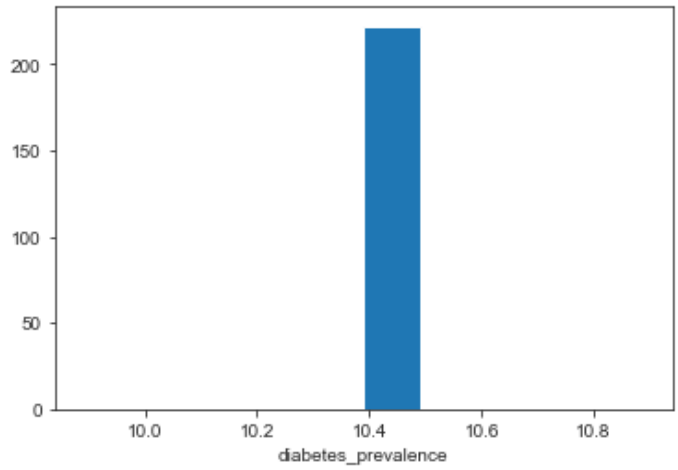



0
0
5
10
15
20
25
30
total_deaths_per_million

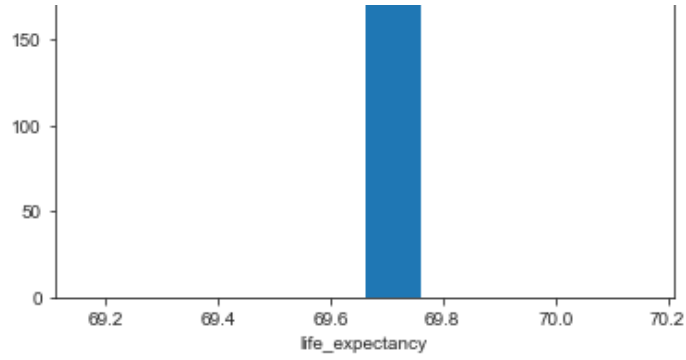

100
80
60
40
20
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
new_tests_smoothed_per_thousand

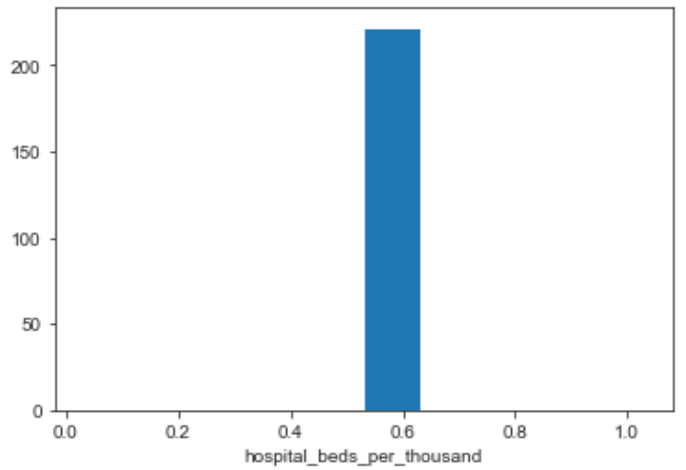

80
60
40
20
0
0.02
0.04
0.06
0.08
0.10
0.12
positive_rate


80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
stringency_index


200
150
100
50
0
10.0
10.2
10.4
10.6
10.8
diabetes_prevalence


200





2: Calculating the Mean,Mode and Median of all the columns

In [252]:

```
float_col=['total_cases','new_cases','total_deaths','new_deaths','total_cases_per_million
','new_cases_per_million','total_deaths_per_million','new_deaths_per_million','new_tests'
,'total_tests','total_tests_per_thousand','new_tests_per_thousand','new_tests_smoothed','
new_tests_smoothed_per_thousand','tests_per_case','positive_rate','stringency_index','pop
ulation','population_density','median_age','aged_65_older','aged_70_older','gdp_per_capit
a','extreme_poverty','cardiovasc_death_rate','diabetes_prevalence','female_smokers','male
_smokers','handwashing_facilities','hospital_beds_per_thousand','life_expectancy']
for col in float_col:
    print("mean of "+col+" is ",df[col].mean())
# mean of all the columns
```

```
mean of total_cases is  283307.9504504505
mean of new_cases is  9698.243243243243
mean of total_deaths is  7095.31981981982
mean of new_deaths is  195.4009009009009
mean of total_cases_per_million is  205.29490990990988
mean of new_cases_per_million is  7.0276846846846865
mean of total_deaths_per_million is  5.141522522522522
mean of new_deaths_per_million is  0.14158558558558562
mean of new_tests is  174257.73076923066
mean of total_tests is  5955751.522058826
mean of total_tests_per_thousand is  4.315772058823533
mean of new_tests_per_thousand is  0.12629230769230768
mean of new_tests_smoothed is  148528.9078014184
mean of new_tests_smoothed_per_thousand is  0.1076241134751773
mean of tests_per_case is  18.192177304964545
mean of positive_rate is  0.06600709219858164
mean of stringency_index is  58.258082191780815
mean of population is  1380004385.0
mean of population_density is  450.41899999999885
mean of median_age is  28.19999999999989
mean of aged_65_older is  5.989000000000015
mean of aged_70_older is  3.41399999999999
mean of gdp_per_capita is  6426.674000000031
mean of extreme_poverty is  21.199999999999914
mean of cardiovasc death rate is  282.2799999999992
```

```
mean of diabetes_prevalence is  10.390000000000025
mean of female_smokers is  1.8999999999999935
mean of male_smokers is  20.599999999999984
mean of handwashing_facilities is  59.549999999999805
mean of hospital_beds_per_thousand is  0.5300000000000009
mean of life_expectancy is  69.65999999999988
```

In [253]:

```
float_col=['total_cases','new_cases','total_deaths','new_deaths','total_cases_per_million
','new_cases_per_million','total_deaths_per_million','new_deaths_per_million','new_tests'
,'total_tests','total_tests_per_thousand','new_tests_per_thousand','new_tests_smoothed','
new_tests_smoothed_per_thousand','tests_per_case','positive_rate','stringency_index','pop
ulation','population_density','median_age','aged_65_older','aged_70_older','gdp_per_capit
a','extreme_poverty','cardiovasc_death_rate','diabetes_prevalence','female_smokers','male
_smokers','handwashing_facilities','hospital_beds_per_thousand','life_expectancy']
for col in float_col:
    print("mode of "+col+" is ",df[col].mode())
 # mode of all the columns
```

```
mode of total_cases is  0    0.0
dtype: float64
mode of new_cases is  0    0.0
dtype: float64
mode of total_deaths is  0    0.0
dtype: float64
mode of new_deaths is  0    0.0
dtype: float64
mode of total_cases_per_million is  0    0.0
dtype: float64
mode of new_cases_per_million is  0    0.0
dtype: float64
mode of total_deaths_per_million is  0    0.0
dtype: float64
mode of new_deaths_per_million is  0    0.0
dtype: float64
mode of new_tests is  0    174257.730769
dtype: float64
mode of total_tests is  0    5.955752e+06
dtype: float64
mode of total_tests_per_thousand is  0    4.315772
dtype: float64
mode of new_tests_per_thousand is  0    0.126292
dtype: float64
mode of new_tests_smoothed is  0    148528.907801
dtype: float64
mode of new_tests_smoothed_per_thousand is  0    0.107624
dtype: float64
mode of tests_per_case is  0    18.192177
dtype: float64
mode of positive_rate is  0    0.066007
dtype: float64
mode of stringency_index is  0    10.19
dtype: float64
mode of population is  0    1.380004e+09
dtype: float64
mode of population_density is  0    450.419
dtype: float64
mode of median_age is  0    28.2
dtype: float64
mode of aged_65_older is  0    5.989
dtype: float64
mode of aged_70_older is  0    3.414
dtype: float64
mode of gdp_per_capita is  0    6426.674
dtype: float64
mode of extreme_poverty is  0    21.2
dtype: float64
mode of cardiovasc_death_rate is  0    282.28
dtype: float64
mode of diabetes_prevalence is  0    10.39
dtype: float64
```

```
mode of female_smokers is  0    1.9
dtype: float64
mode of male_smokers is  0    20.6
dtype: float64
mode of handwashing_facilities is  0    59.55
dtype: float64
mode of hospital_beds_per_thousand is  0    0.53
dtype: float64
mode of life_expectancy is  0    69.66
dtype: float64
```

In [254]:

```
float_col=['total_cases','new_cases','total_deaths','new_deaths','total_cases_per_million
','new_cases_per_million','total_deaths_per_million','new_deaths_per_million','new_tests'
,'total_tests','total_tests_per_thousand','new_tests_per_thousand','new_tests_smoothed','
new_tests_smoothed_per_thousand','tests_per_case','positive_rate','stringency_index','pop
ulation','population_density','median_age','aged_65_older','aged_70_older','gdp_per_capit
a','extreme_poverty','cardiovasc_death_rate','diabetes_prevalence','female_smokers','male
_smokers','handwashing_facilities','hospital_beds_per_thousand','life_expectancy']
for col in float_col:
    print("median of "+col+" is ",df[col].median())
 # median of all the columns
```

```
median of total_cases is  17932.5
median of new_cases is  1359.5
median of total_deaths is  566.5
median of new_deaths is  40.5
median of total_cases_per_million is  12.994499999999999
median of new_cases_per_million is  0.985
median of total_deaths_per_million is  0.4105
median of new_deaths_per_million is  0.0295
median of new_tests is  174257.73076923078
median of total_tests is  5955751.522058823
median of total_tests_per_thousand is  4.315772058823529
median of new_tests_per_thousand is  0.12629230769230768
median of new_tests_smoothed is  148528.90780141845
median of new_tests_smoothed_per_thousand is  0.10762411347517736
median of tests_per_case is  18.192177304964545
median of positive_rate is  0.0660070921985816
median of stringency_index is  76.39
median of population is  1380004385.0
median of population_density is  450.41900000000004
median of median_age is  28.2
median of aged_65_older is  5.989
median of aged_70_older is  3.4139999999999997
median of gdp_per_capita is  6426.674
median of extreme_poverty is  21.2
median of cardiovasc_death_rate is  282.28
median of diabetes_prevalence is  10.39
median of female_smokers is  1.9
median of male_smokers is  20.6
median of handwashing_facilities is  59.55
median of hospital_beds_per_thousand is  0.53
median of life_expectancy is  69.66
```

Bivariate Analysis

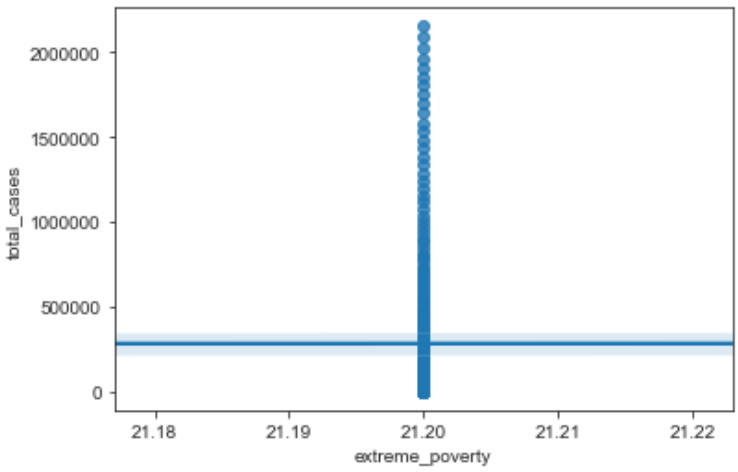
1: Scatterplots

In [255]:

```
import seaborn as sns
sns.set_style('white')
sns.set_style('ticks')
sns.regplot(x='extreme_poverty',y='total_cases',data=df)
```

Out[255]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d3234a9518>

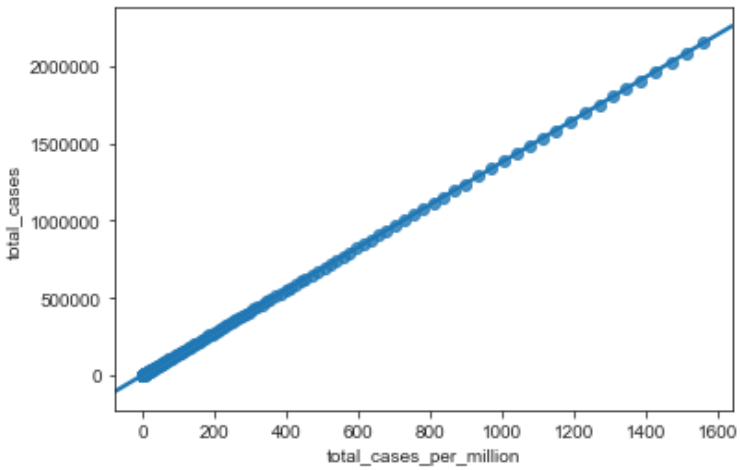


In [256]:

```
sns.set_style('white')
sns.set_style('ticks')
sns.regplot(x='total_cases_per_million',y='total_cases',data=df)
```

Out[256]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d323506400>

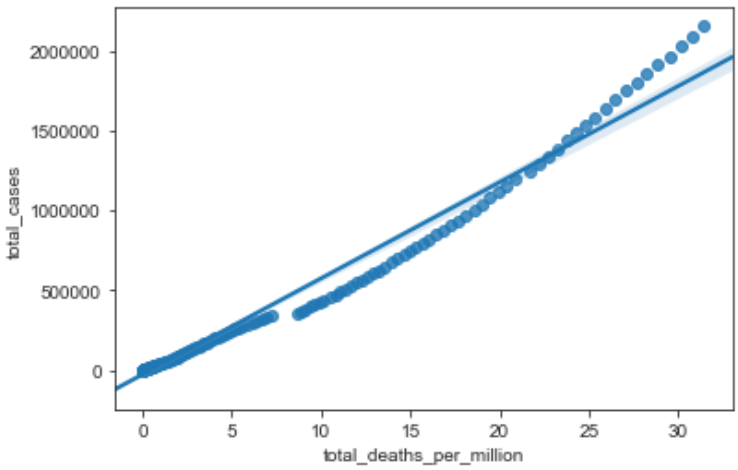


In [257]:

```
sns.regplot(x='total_deaths_per_million',y='total_cases',data=df)
```

Out[257]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d323572be0>

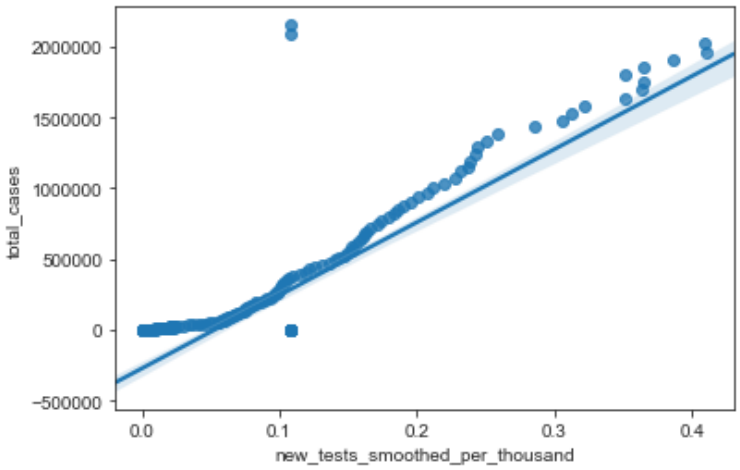


In [258]:

```
sns.regplot(x='new_tests_smoothed_per_thousand',y='total_cases',data=df)
```

Out[258]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d3235dc550>

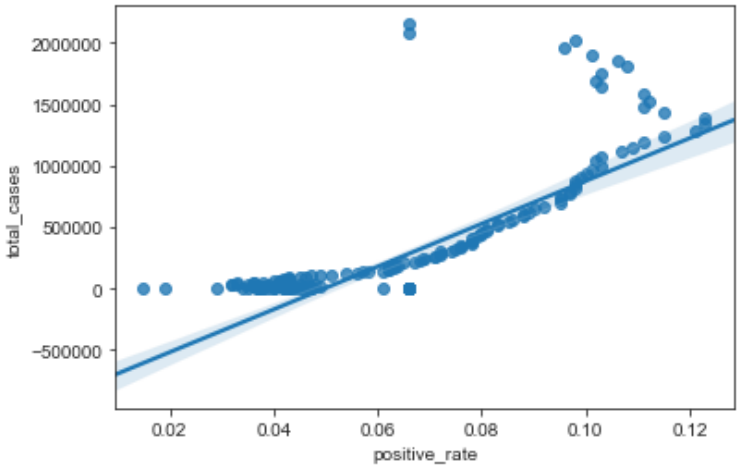


In [259]:

```
sns.regplot(x='positive_rate',y='total_cases',data=df)
```

Out[259]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d32363b898>

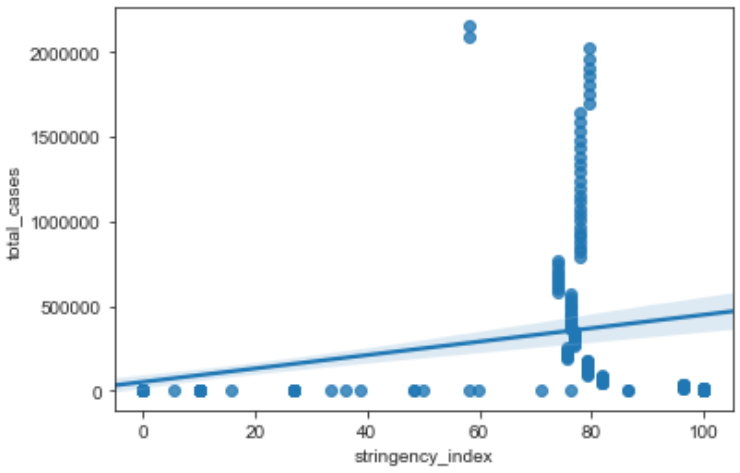


In [260]:

```
sns.regplot(x='stringency_index',y='total_cases',data=df)
```

Out[260]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d3236a03c8>

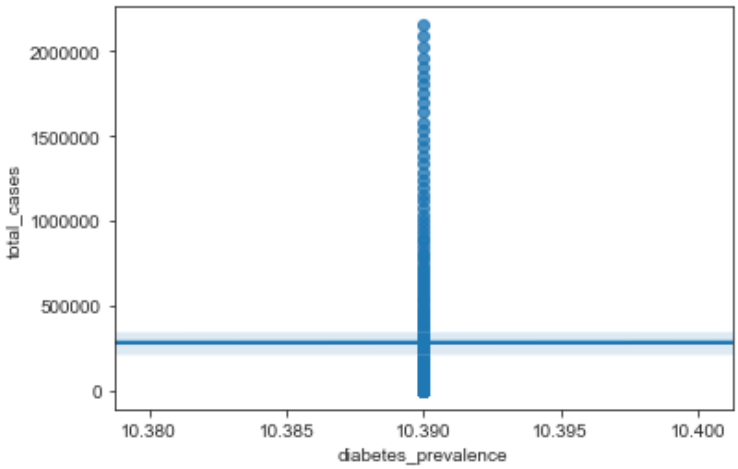


In [261]:

```
sns.regplot(x='diabetes_prevalence',y='total_cases',data=df)
```

Out[261]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d3237026d8>

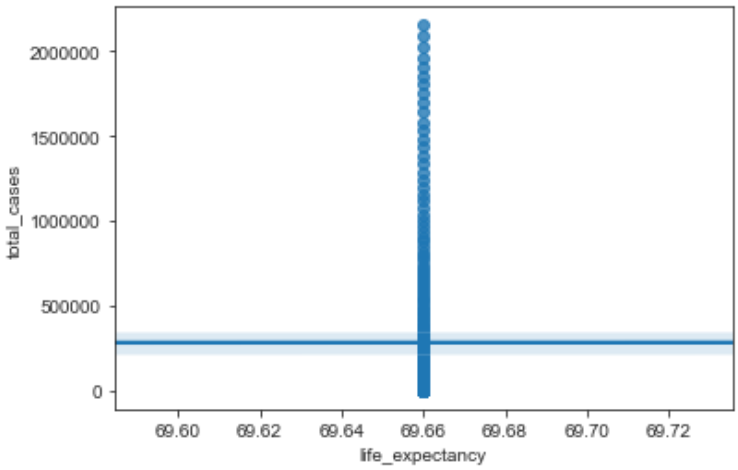


In [262]:

```
sns.regplot(x='life_expectancy',y='total_cases',data=df)
```

Out[262]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d323763a90>

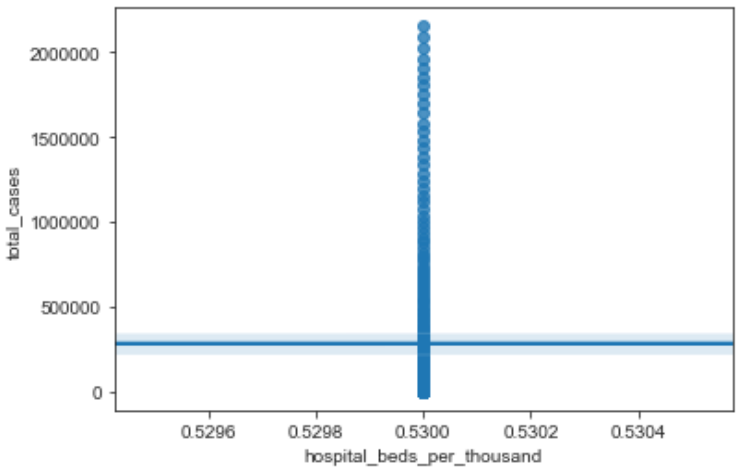


In [263]:

```
sns.regplot(x='hospital_beds_per_thousand',y='total_cases',data=df)
```

Out[263]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d3237c72b0>



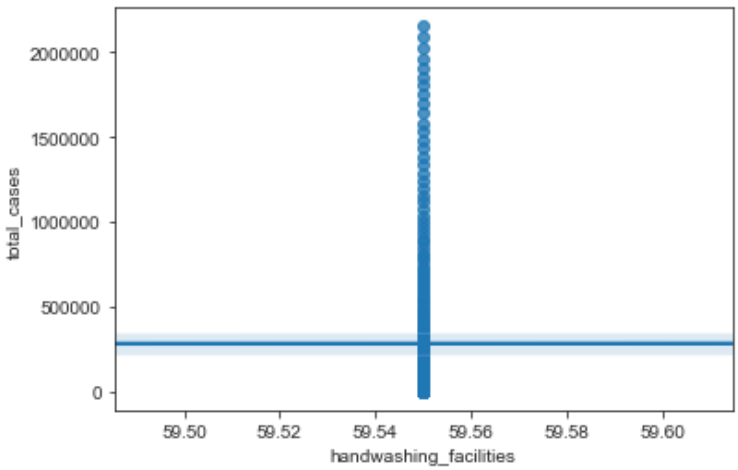
In [264]:

```
sns.regplot(x='handwashing_facilities',y='total_cases',data=df)
```

Out[264]:

Out[264]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d32381bbe0>
```



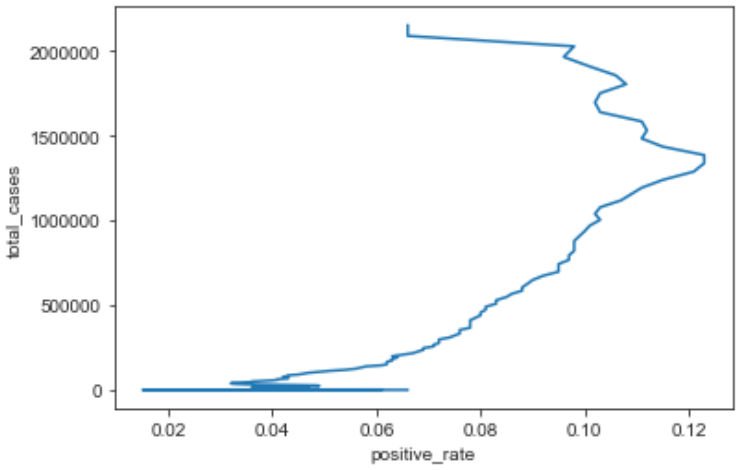
2: Lineplots

In [265]:

```
graphs=["positive_rate","handwashing_facilities","hospital_beds_per_thousand","stringency
_index","new_tests_smoothed_per_thousand","total_deaths_per_million","total_cases_per_mil
lion","female_smokers","aged_70_older","extreme_poverty"]
plt.plot("positive_rate","total_cases",data=df)
plt.xlabel("positive_rate")
plt.ylabel("total_cases")
```

Out[265]:

```
Text(0, 0.5, 'total_cases')
```

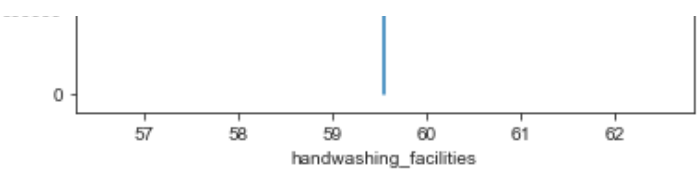


In [266]:

```
plt.plot("handwashing_facilities","total_cases",data=df)
plt.xlabel("handwashing_facilities")
plt.ylabel("total_cases")
```

Out[266]:

```
Text(0, 0.5, 'total_cases')
```

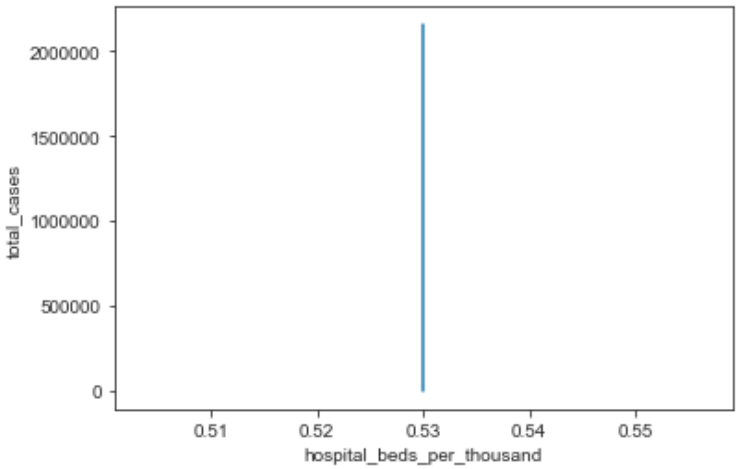


In [267]:

```
plt.plot("hospital_beds_per_thousand","total_cases",data=df)
plt.xlabel("hospital_beds_per_thousand")
plt.ylabel("total_cases")
```

Out[267]:

```
Text(0, 0.5, 'total_cases')
```

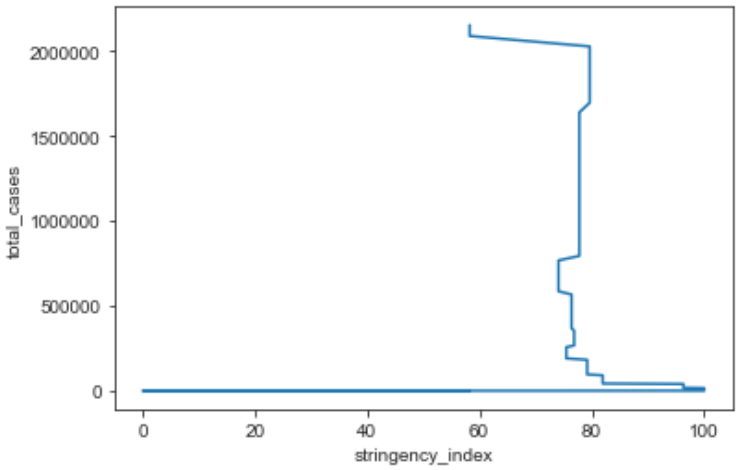


In [268]:

```
plt.plot("stringency_index","total_cases",data=df)
plt.xlabel("stringency_index")
plt.ylabel("total_cases")
```

Out[268]:

```
Text(0, 0.5, 'total_cases')
```

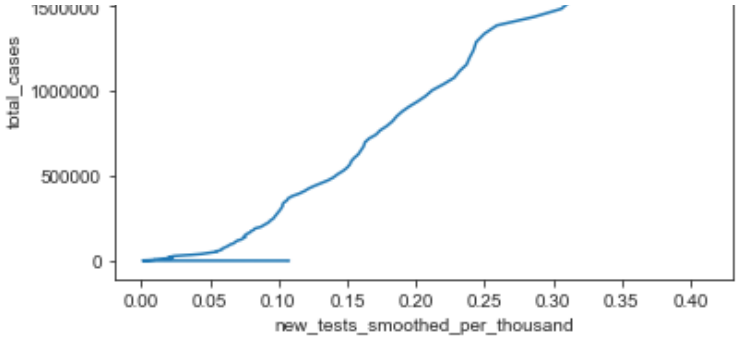


In [269]:

```
plt.plot("new_tests_smoothed_per_thousand","total_cases",data=df)
plt.xlabel("new_tests_smoothed_per_thousand")
plt.ylabel("total_cases")
```

Out[269]:

```
Text(0, 0.5, 'total_cases')
```

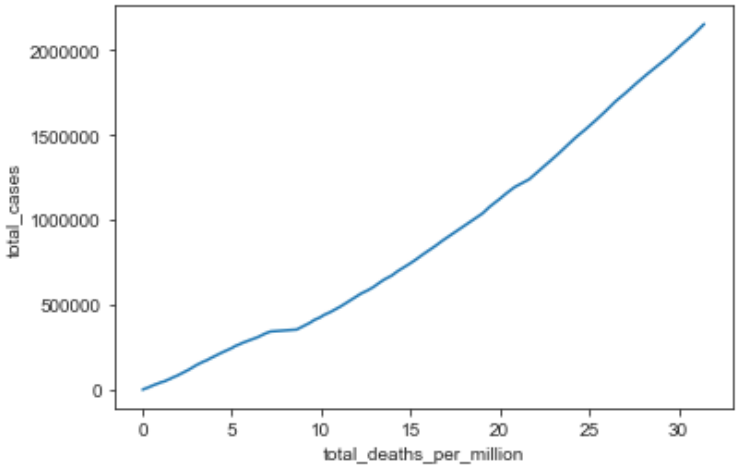


In [270]:

```
plt.plot("total_deaths_per_million","total_cases",data=df)
plt.xlabel("total_deaths_per_million")
plt.ylabel("total_cases")
```

Out[270]:

```
Text(0, 0.5, 'total_cases')
```

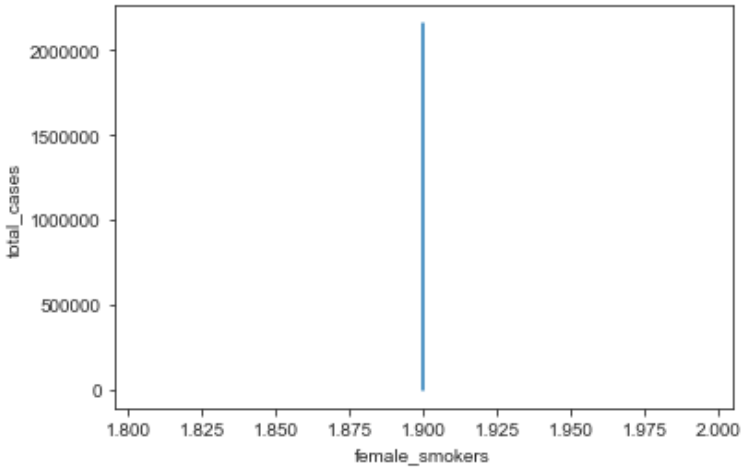


In [271]:

```
plt.plot("female_smokers","total_cases",data=df)
plt.xlabel("female_smokers")
plt.ylabel("total_cases")
```

Out[271]:

```
Text(0, 0.5, 'total_cases')
```



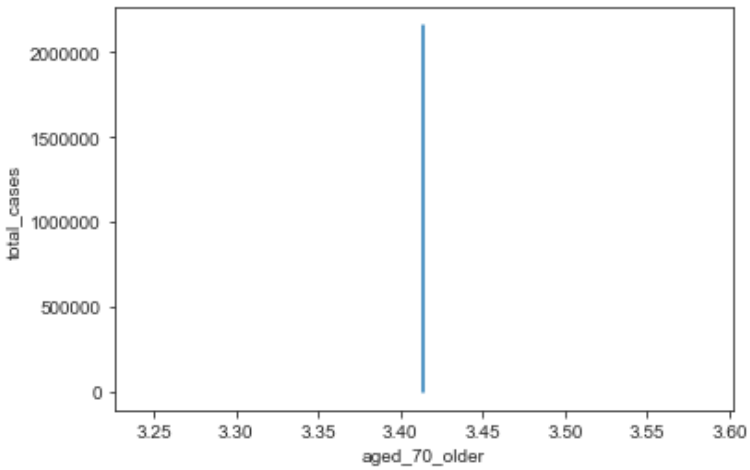
In [272]:

```
plt.plot("aged_70_older","total_cases",data=df)
plt.xlabel("aged_70_older")
plt.ylabel("total_cases")
```

Out[272]:

```
Text(0, 0.5, 'total_cases')
```

Text(0, 0.5, 'total_cases')

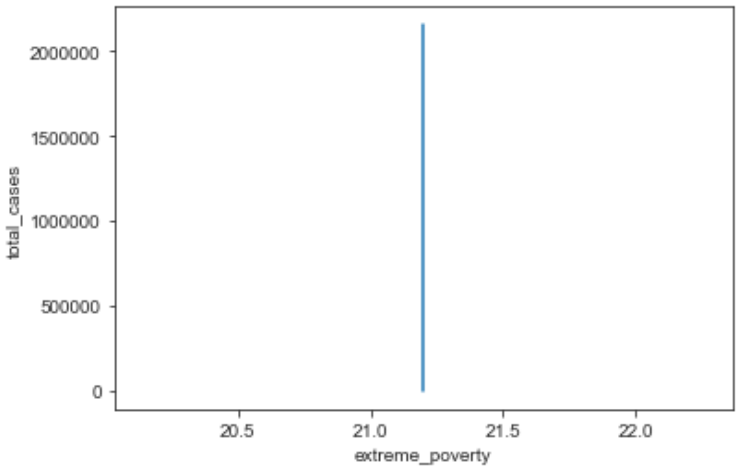


In [273]:

```
plt.plot("extreme_poverty","total_cases",data=df)
plt.xlabel("extreme_poverty")
plt.ylabel("total_cases")
```

Out[273]:

```
Text(0, 0.5, 'total_cases')
```

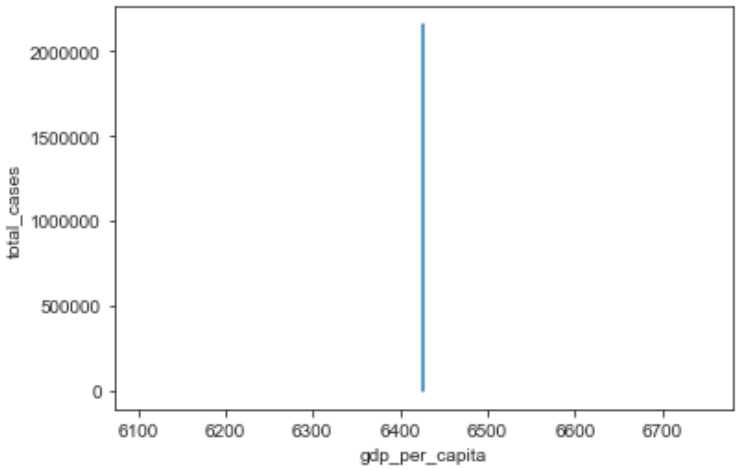


In [274]:

```
plt.plot("gdp_per_capita","total_cases",data=df)
plt.xlabel("gdp_per_capita")
plt.ylabel("total_cases")
```

Out[274]:

```
Text(0, 0.5, 'total_cases')
```



Converting the date column into ordinal

Converting the date column into ordinal

In [275]:

```
import datetime as dt
```

In [276]:

```
df["date"]=pd.to_datetime(df["date"])
df["date"]=df["date"].map(dt.datetime.toordinal)
```

In [277]:

```
df.head() #for checking the date column to confirm
```

Out[277]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_
14956	**IND**	**Asia**	**India**	**737424**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	
14957	**IND**	**Asia**	**India**	**737425**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	
14958	**IND**	**Asia**	**India**	**737426**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	
14959	**IND**	**Asia**	**India**	**737427**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	
14960	**IND**	**Asia**	**India**	**737428**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	

5 rows × 36 columns

Dropping the useless categoical columns and converting the useful ones into ordinal using Label Encoder

In [278]:

```
useless=["iso_code","continent","location"]
df=df.drop(useless,axis=1)  # dropping all the useless categorical columns
```

In [279]:

```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
```

In [280]:

```
label.fit(df["tests_units"])
```

Out[280]:

```
LabelEncoder()
```

In [281]:

```
df["tests_units"]=label.transform(df["tests_units"])  # converting into ordinal using LabelEncoder
```

In [282]:

```
df["tests_units"].head() # confirming whether the values are converted
```

Out[282]:

```
14956    0
14957    0
14958    0
14959    0
14960    0
Name: tests_units, dtype: int32
```

In [283]:

```
x=df.drop("total_cases",axis=1)  # feature variable
```

```
x=df.drop("total_cases",axis=1)  # feature variable
y=df["total_cases"] # target variable
```

Performing train-test split

In [284]:

```
from sklearn.model_selection import train_test_split # Splitting train test
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

Modelling:

1: Linear Regression

In [285]:

```
from sklearn.linear_model import LinearRegression
```

In [286]:

```
lreg=LinearRegression()
```

In [287]:

```
lreg.fit(x_train,y_train)
```

Out[287]:

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

In [288]:

```
lreg.score(x_test,y_test)  # accuracy on the testing dataset
```

Out[288]:

```
0.9999999999991778
```

In [289]:

```
lreg.predict(x_test)
```

Out[289]:

```
array([2.07614981e+05, 1.80369434e+06, 6.45414157e-02, 4.25281340e+05,
       2.46627699e+05, 4.64330117e+04, 4.42129063e+03, 8.28917602e-02,
       3.00047403e+00, 3.46438655e+01, 4.93905091e+04, 2.97534503e+05,
       1.18447254e+05, 2.45061448e+04, 2.92045453e+00, 3.25336177e-02,
       6.41118092e+03, 4.06655311e+03, 8.05453147e-02, 4.73105488e+05,
       4.38798279e+02, 1.51495207e+00, 7.67296564e+05, 2.92022613e+01,
       1.85998300e+04, 2.97380086e+00, 3.04439890e+01, 3.80532128e+05,
       2.94356876e+04, 4.42999682e+01, 2.30034499e+03, 6.98760487e-02,
       1.57114461e+04, 2.16918955e+05, 1.99840385e+04, 1.25101626e+05,
       9.68876401e+05, 1.58379259e+06, 1.91353576e+02, 1.01139192e+05,
       1.06750156e+05, 3.13315778e+04, 2.02707482e+06, 5.61719799e+02,
       4.25332611e+04, 8.29965084e+01, 1.73763521e+05, 3.37391053e+03,
       2.78928595e+04, 9.24910377e+01, 7.44642796e+03, 1.45379311e+05,
       1.12358545e+05, 2.88311210e+00, 1.12553113e-01, 5.48318447e+05,
       5.66839756e+05, 9.06752548e+05, 5.29518618e+04, 7.50879518e-02,
       8.58799477e-02, 3.08992648e+05, 1.00383165e+06, 1.33686123e+06,
       2.13930983e+04, 1.11950857e-02, 2.97913549e+00])
```

2: Random Forest Regressor

In [290]:

```
from sklearn.ensemble import RandomForestRegressor
ran=RandomForestRegressor()
```

In [291]:

```
ran.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning:
The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[291]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

In [292]:

```
ran.score(x_test,y_test) # accuracy on the testing dataset
```

Out[292]:

```
0.9989530701392547
```

In [293]:

```
ran.predict(x_test)
```

Out[293]:

```
array([2.0415580e+05, 1.7916035e+06, 0.0000000e+00, 4.1972140e+05,
       2.4317800e+05, 5.5968900e+04, 3.6967000e+03, 0.0000000e+00,
       3.0000000e+00, 2.3100000e+01, 5.4721000e+04, 3.2006910e+05,
       1.2689160e+05, 2.1195200e+04, 3.0000000e+00, 0.0000000e+00,
       6.1214000e+03, 5.3743000e+03, 0.0000000e+00, 4.6827340e+05,
       4.4200000e+02, 8.0000000e-01, 7.4694310e+05, 1.4300000e+01,
       1.7802600e+04, 3.0000000e+00, 1.4200000e+01, 3.7733630e+05,
       2.5314300e+04, 5.0500000e+01, 1.5555000e+03, 0.0000000e+00,
       1.4647500e+04, 2.2463400e+05, 1.9134500e+04, 1.3636870e+05,
       9.8247920e+05, 1.5638293e+06, 2.5940000e+02, 9.6024500e+04,
       1.0193520e+05, 3.2432200e+04, 1.9439289e+06, 5.2560000e+02,
       3.9451200e+04, 8.2500000e+01, 1.8419800e+05, 2.0508000e+03,
       2.0853300e+04, 8.8000000e+01, 9.8121000e+03, 1.4298720e+05,
       1.1062210e+05, 3.0000000e+00, 0.0000000e+00, 5.1626060e+05,
       5.7809550e+05, 8.8989190e+05, 5.9224800e+04, 0.0000000e+00,
       0.0000000e+00, 3.4468760e+05, 1.0436605e+06, 1.3269018e+06,
       1.8429700e+04, 0.0000000e+00, 3.0000000e+00])
```

Predicting the Total Case on a new sample test case

In [294]:

```
sample=pd.read_csv("covidonlyfeatures.csv")
```

In [295]:

```
sample
```

Out[295]:

	date	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million	total_deaths_per_million
0	**08/12/2020**	**60963**	**46091**	**834**	**1688.14**	**44.18**	**33.4**

1 rows × 32 columns

In [296]:

```
sample["date"]=pd.to_datetime(sample["date"])
sample["date"]=sample["date"].map(dt.datetime.toordinal)
```

In [297]:

```
sample  # confirming whether the date is converted
```

Out[297]:

	date	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million	total_deaths_per_million	new
0	**737649**	**60963**	**46091**	**834**	**1688.14**	**44.18**	**33.4**	

1 rows × 32 columns

In [298]:

```
label.fit(sample["tests_units"])  # using LabelEncoder for converting the categorical col
umn into ordinal
```

Out[298]:

```
LabelEncoder()
```

In [299]:

```
sample["tests_units"]=label.transform(sample["tests_units"])
```

In [300]:

```
sample["tests_units"] # confirming
```

Out[300]:

```
0    0
Name: tests_units, dtype: int64
```

Predicting using Linear Regression

In [301]:

```
lreg.predict(sample)
```

Out[301]:

```
array([2325470.27400979])
```

Predicting using Random Forest Regressor

In [302]:

```
ran.predict(sample)
```

Out[302]:

```
array([1968440.8])
```

In [ ]:

```

```