In [1]:
```
#Importing Libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Load the Dataset

CountryPopulationGrowthPredictions= pd.read_csv('CountryPopulationGrowthPredictions
CountryPopulationGrowthPredictions.head()
```

Out[2]:

	Country	Area_km	Population	Pop_Density	Yearly_Change	One_Year_Prediction	Density_On
0	Macao	30	649335	21645	1.39	658361	
1	Singapore	700	5850342	8358	0.79	5896560	
2	Hong Kong	1050	7496981	7140	0.82	7558456	
3	Bahrain	760	1701575	2239	3.68	1764193	
4	Maldives	300	540544	1802	1.81	550328	

In [3]:
```
#Describe Function

CountryPopulationGrowthPredictions.describe(include='all')
```

Out[3]:

	Country	Area_km	Population	Pop_Density	Yearly_Change	One_Year_Prediction
count	201	2.010000e+02	2.010000e+02	201.000000	201.000000	2.010000e+02
unique	201	NaN	NaN	NaN	NaN	NaN
top	Macao	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN
mean	NaN	6.450903e+05	3.877661e+07	361.711443	1.200299	3.918842e+07
std	NaN	1.809408e+06	1.454245e+08	1710.321831	1.091574	1.464647e+08
min	NaN	3.000000e+01	9.792900e+04	2.000000	-2.470000	9.875200e+04
25%	NaN	2.164000e+04	1.886198e+06	34.000000	0.420000	1.865827e+06
50%	NaN	1.085600e+05	8.654622e+06	89.000000	1.080000	8.702422e+06
75%	NaN	4.988000e+05	2.769102e+07	228.000000	1.960000	2.835632e+07
max	NaN	1.637687e+07	1.439324e+09	21645.000000	3.840000	1.444937e+09

In [4]:
```
CountryPopulationGrowthPredictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Country                      201 non-null    object 
 1   Area_km                      201 non-null    int64  
 2   Population                   201 non-null    int64  
 3   Pop_Density                  201 non-null    int64  
 4   Yearly_Change                201 non-null    float64
 5   One_Year_Prediction          201 non-null    int64  
 6   Density_One_Year             201 non-null    int64  
 7   Ten_Year_Prediction          201 non-null    int64  
 8   Density_Ten_Year             201 non-null    int64  
 9   One_Hundred_Year_Prediction  201 non-null    int64  
 10  Density_One_Hundred_Year     201 non-null    int64  
dtypes: float64(1), int64(9), object(1)
memory usage: 17.4+ KB
```

In [5]:

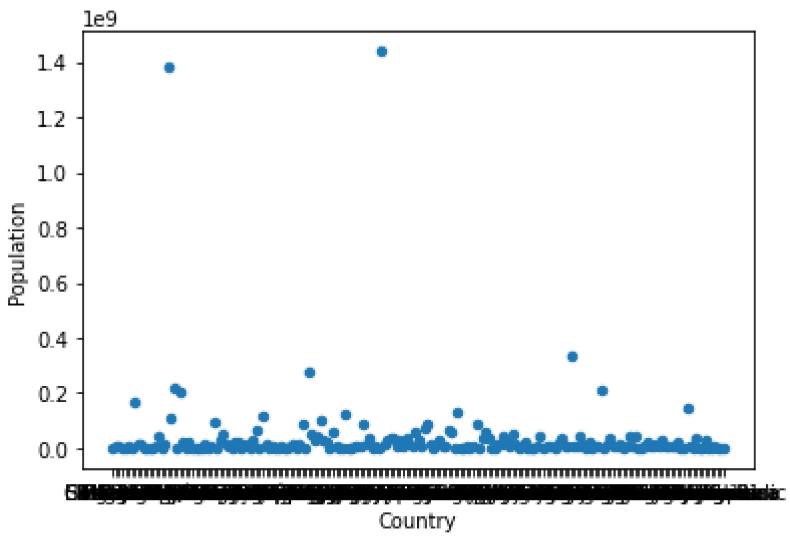
```
CountryPopulationGrowthPredictions.isna().sum()
```

Out[5]:

```
Country                        0
Area_km                        0
Population                     0
Pop_Density                    0
Yearly_Change                  0
One_Year_Prediction            0
Density_One_Year               0
Ten_Year_Prediction            0
Density_Ten_Year               0
One_Hundred_Year_Prediction    0
Density_One_Hundred_Year       0
dtype: int64
```

In [6]:

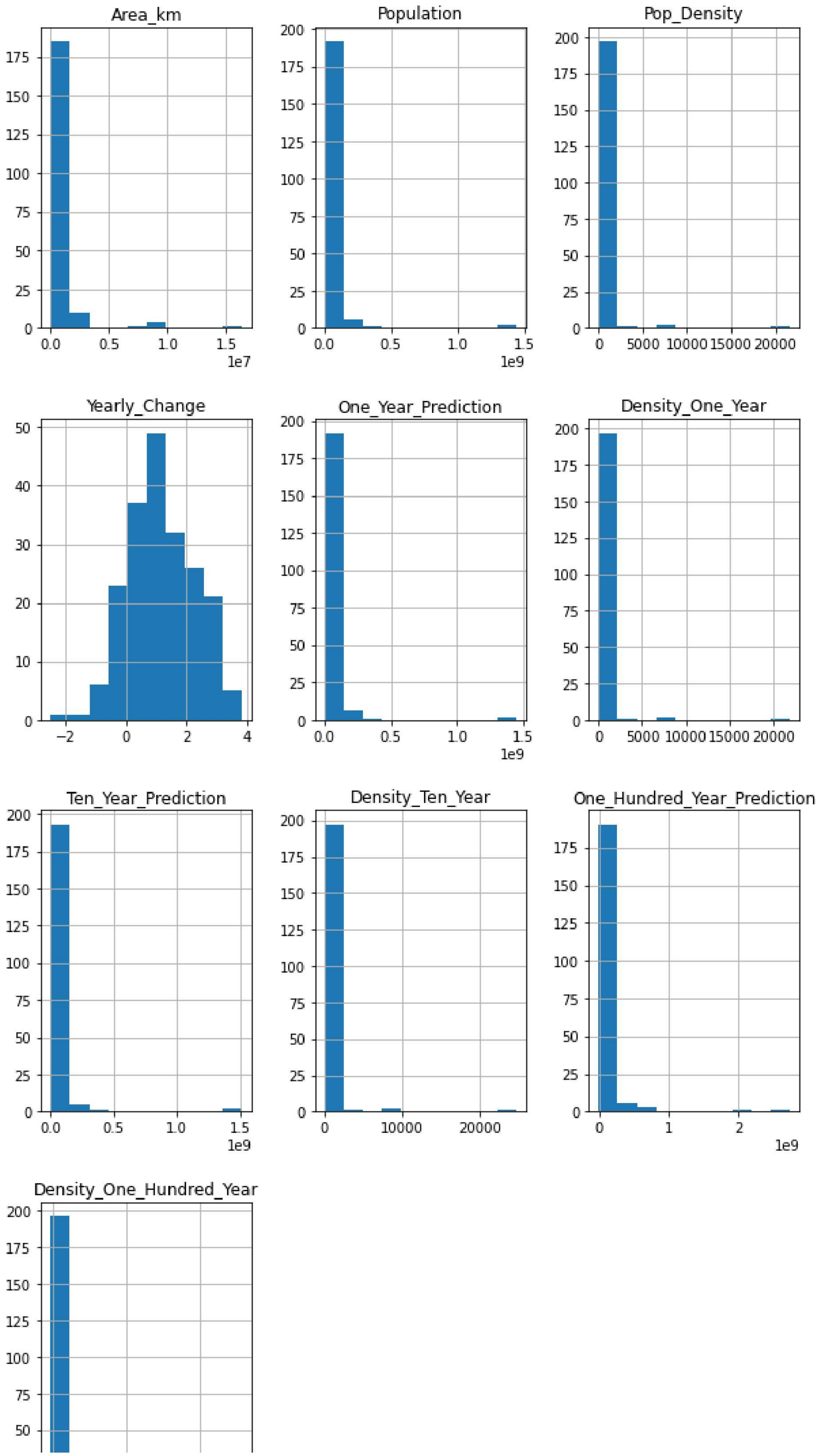
```
# Scatter Plot

CountryPopulationGrowthPredictions.plot(kind='scatter', x='Country', y='Population
plt.show()
```

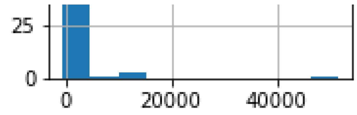


In [7]:

```
# Plot Histogram

CountryPopulationGrowthPredictions.hist(figsize=(10,20))
```

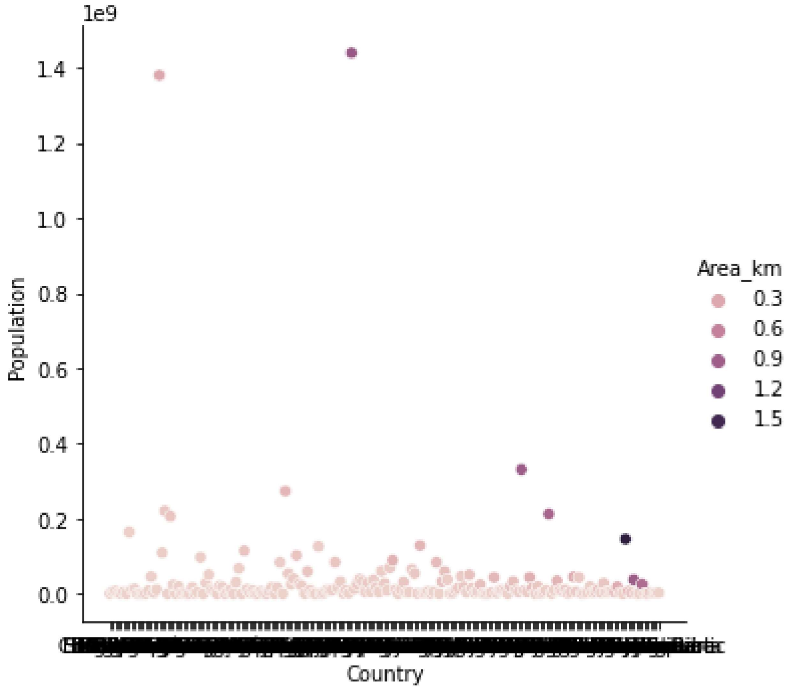
Out[7]:
```
array([[<AxesSubplot:title={'center':'Area_km'}>,
        <AxesSubplot:title={'center':'Population'}>,
        <AxesSubplot:title={'center':'Pop_Density'}>],
       [<AxesSubplot:title={'center':'Yearly_Change'}>,
        <AxesSubplot:title={'center':'One_Year_Prediction'}>,
        <AxesSubplot:title={'center':'Density_One_Year'}>],
       [<AxesSubplot:title={'center':'Ten_Year_Prediction'}>,
        <AxesSubplot:title={'center':'Density_Ten_Year'}>,
        <AxesSubplot:title={'center':'One_Hundred_Year_Prediction'}>],
       [<AxesSubplot:title={'center':'Density_One_Hundred_Year'}>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```


Area_km
0 25 50 75 100 125 150 175
0.0 0.5 1.0 1.5
1e7
Population
0 25 50 75 100 125 150 175 200
0.0 0.5 1.0 1.5
1e9
Pop_Density
0 25 50 75 100 125 150 175 200
0 5000 10000 15000 20000
Yearly_Change
0 10 20 30 40 50
−2 0 2 4
One_Year_Prediction
0 25 50 75 100 125 150 175 200
0.0 0.5 1.0 1.5
1e9
Density_One_Year
0 25 50 75 100 125 150 175 200
0 5000 10000 15000 20000
Ten_Year_Prediction
0 25 50 75 100 125 150 175 200
0.0 0.5 1.0 1.5
1e9
Density_Ten_Year
0 25 50 75 100 125 150 175 200
0 10000 20000
One_Hundred_Year_Prediction
0 25 50 75 100 125 150 175
0 1 2
1e9
Density_One_Hundred_Year
50 75 100 125 150 175 200



In [8]:
```
sns.relplot(x="Country", y="Population", hue="Area_km",data=CountryPopulationGrowth
```

Out[8]:
```
<seaborn.axisgrid.FacetGrid at 0x21cd903f430>
```



In [9]:
```
#seperating independent and dependent variables

X = CountryPopulationGrowthPredictions.drop(['Country'], axis=1)
y = CountryPopulationGrowthPredictions['Population']
X.shape, y.shape
```

Out[9]:
```
((201, 10), (201,))
```

In [10]:
```
# Importing the train test split function and metric mean square error

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error as mae
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_state =
```

In [11]:
```
#Importing Linear Regression

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
model.score(X,y)
```

Out[11]:
```
1.0
```

In [12]:
```
# Predicting over the Train Set and calculating error

train_predict = model.predict(X_train)
k = mae(train_predict, y_train)
print('Training Mean Absolute Error', k )
```

```
Training Mean Absolute Error 2.439273885102011e-08
```

In [13]:
```
# Predicting over the Test Set and calculating error
```

```
test_predict = model.predict(X_test)
k = mae(test_predict, y_test)
print('Test Mean Absolute Error     ', k )
```

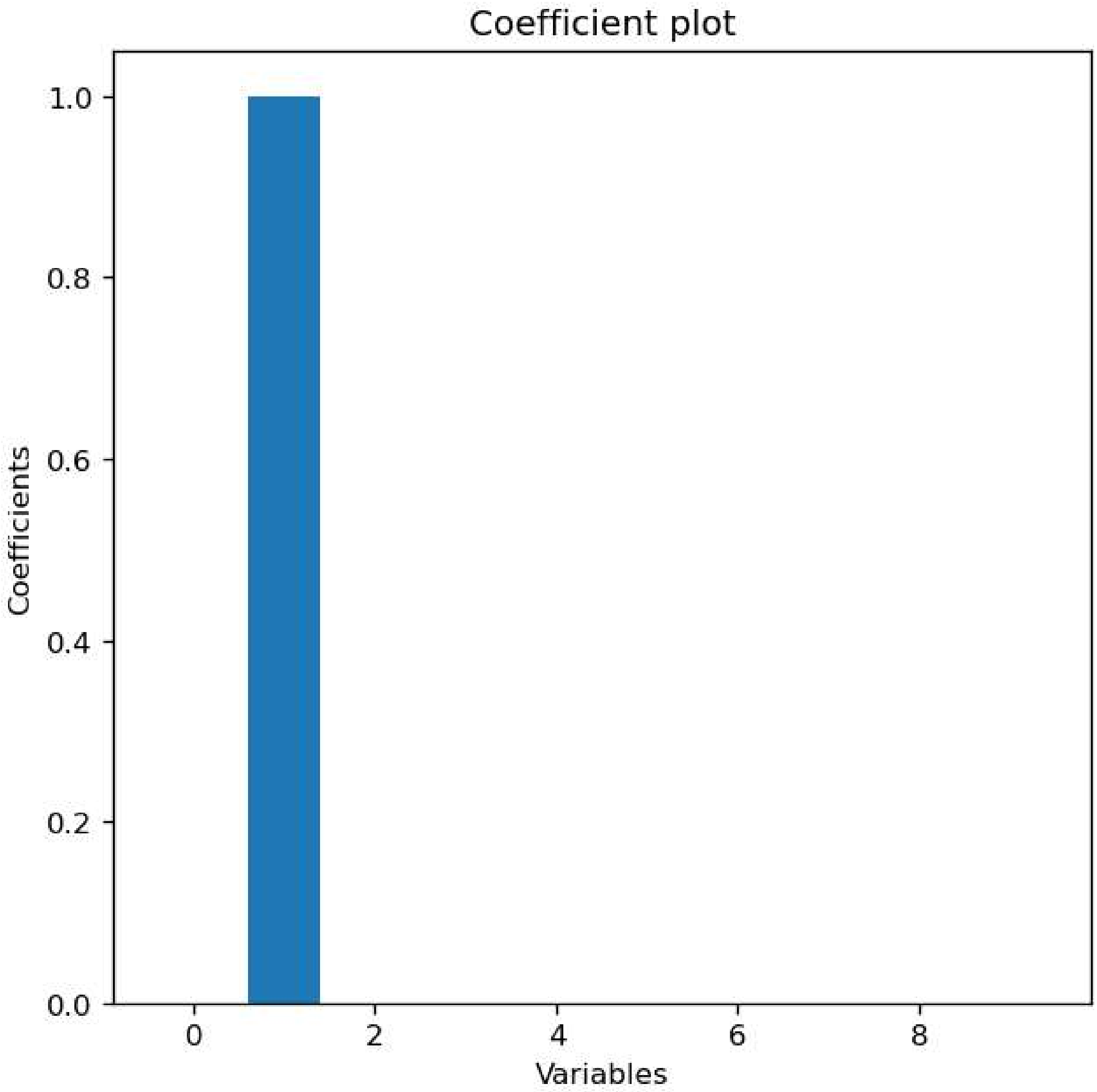
```
Test Mean Absolute Error     3.062397321095554e-08
```

In [14]:
```
#Parameter of Linear Regression

model.coef_
```

Out[14]:
```
array([ 3.96456621e-15,  1.00000000e+00, -3.98944510e-08,  4.85173013e-09,
        1.63811436e-08,  3.43534292e-08, -1.83979877e-08,  6.51841261e-09,
        1.67598679e-09, -9.86463875e-10])
```

In [15]:
```
# Plotting the Cofficients

plt.figure(figsize=(6, 6), dpi=120, facecolor='w', edgecolor='b')
X = range(len(X_train.columns))
y = model.coef_
plt.bar(X, y)
plt.xlabel( "Variables")
plt.ylabel('Coefficients')
plt.title('Coefficient plot')
```

Out[15]:
```
Text(0.5, 1.0, 'Coefficient plot')
```



Checking assumptions of Linear Model

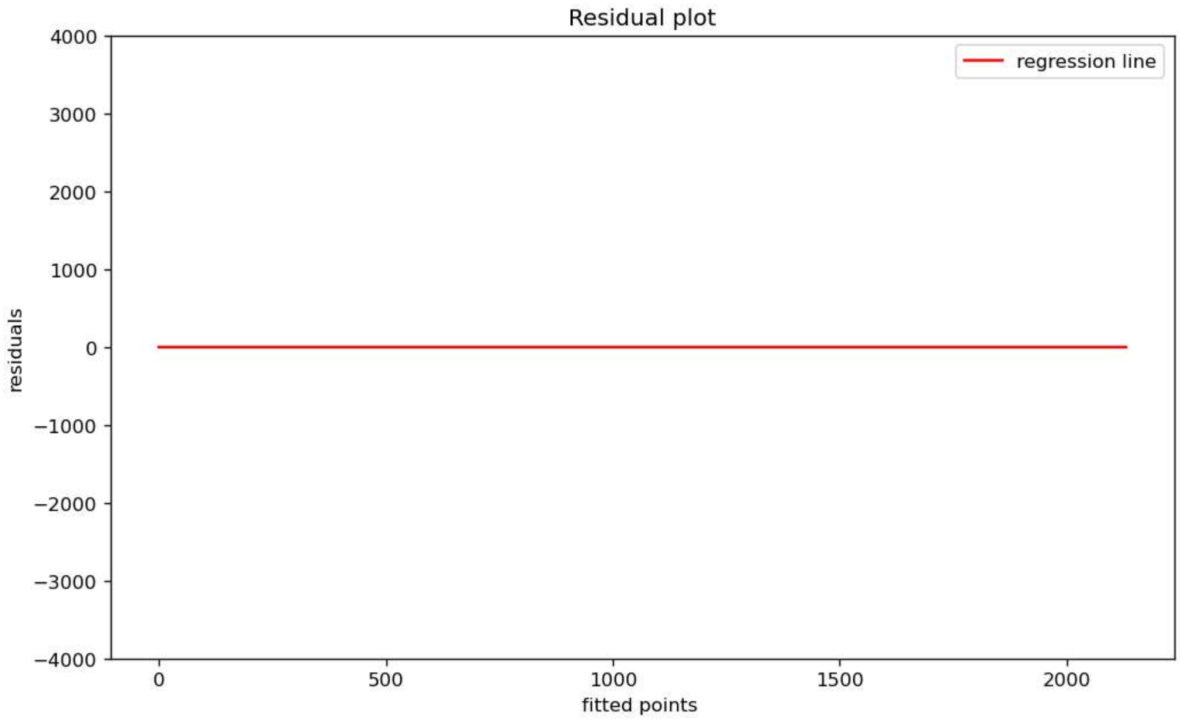
In [16]:
```
# Arranging and calculating the Residuals
residuals = pd.DataFrame({
    'fitted values' : y_test,
    'predicted values' : test_predict,
})

residuals['residuals'] = residuals['fitted values'] - residuals['predicted values']
residuals.head()
```

Out[16]:

	fitted values	predicted values	residuals
17	11402528	11402528.0	1.117587e-08
92	33469203	33469203.0	2.607703e-08
188	145934462	145934462.0	-1.192093e-07
73	9890402	9890402.0	-1.862645e-08
45	183627	183627.0	2.529123e-08

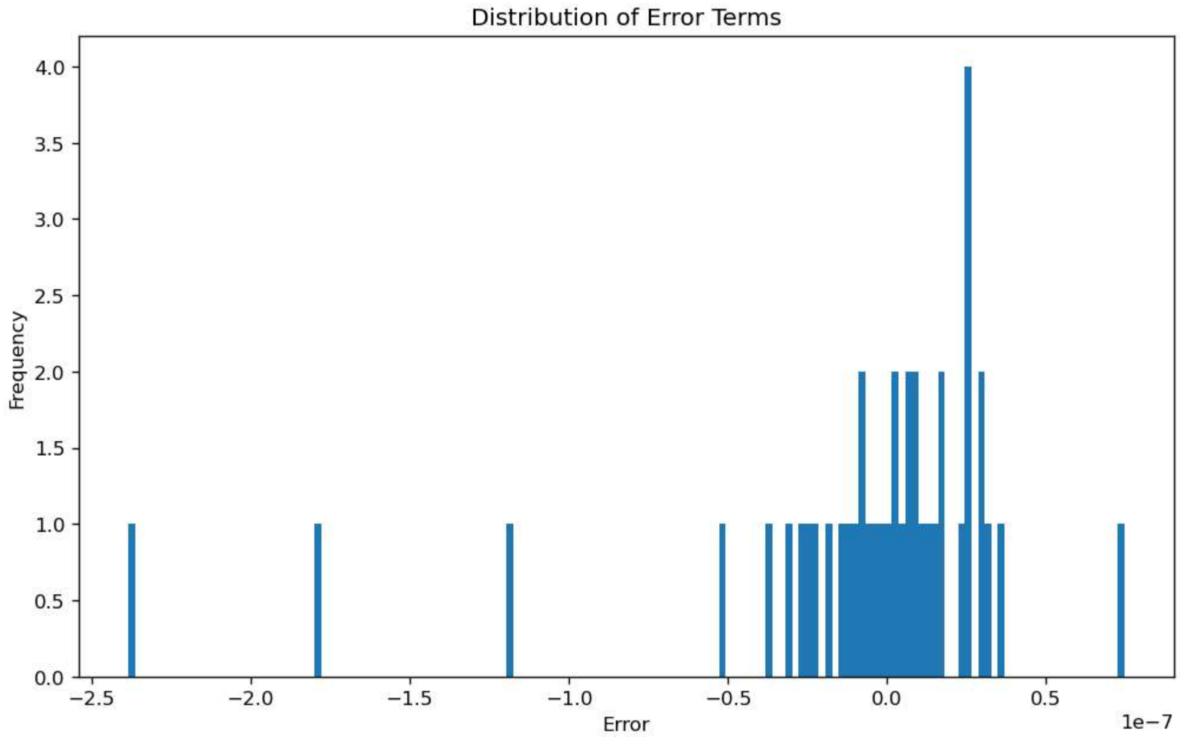
In [21]:
```
plt.figure(figsize=(10, 6), dpi=120, facecolor='w', edgecolor='b')
f = range(0,2131)
k = [0 for i in range(0,2131)]
plt.plot( f, k , color = 'red', label = 'regression line' )
plt.xlabel('fitted points ')
plt.ylabel('residuals')
plt.title('Residual plot')
plt.ylim(-4000, 4000)
plt.legend()
```

Out[21]: `<matplotlib.legend.Legend at 0x21cdb58d820>`



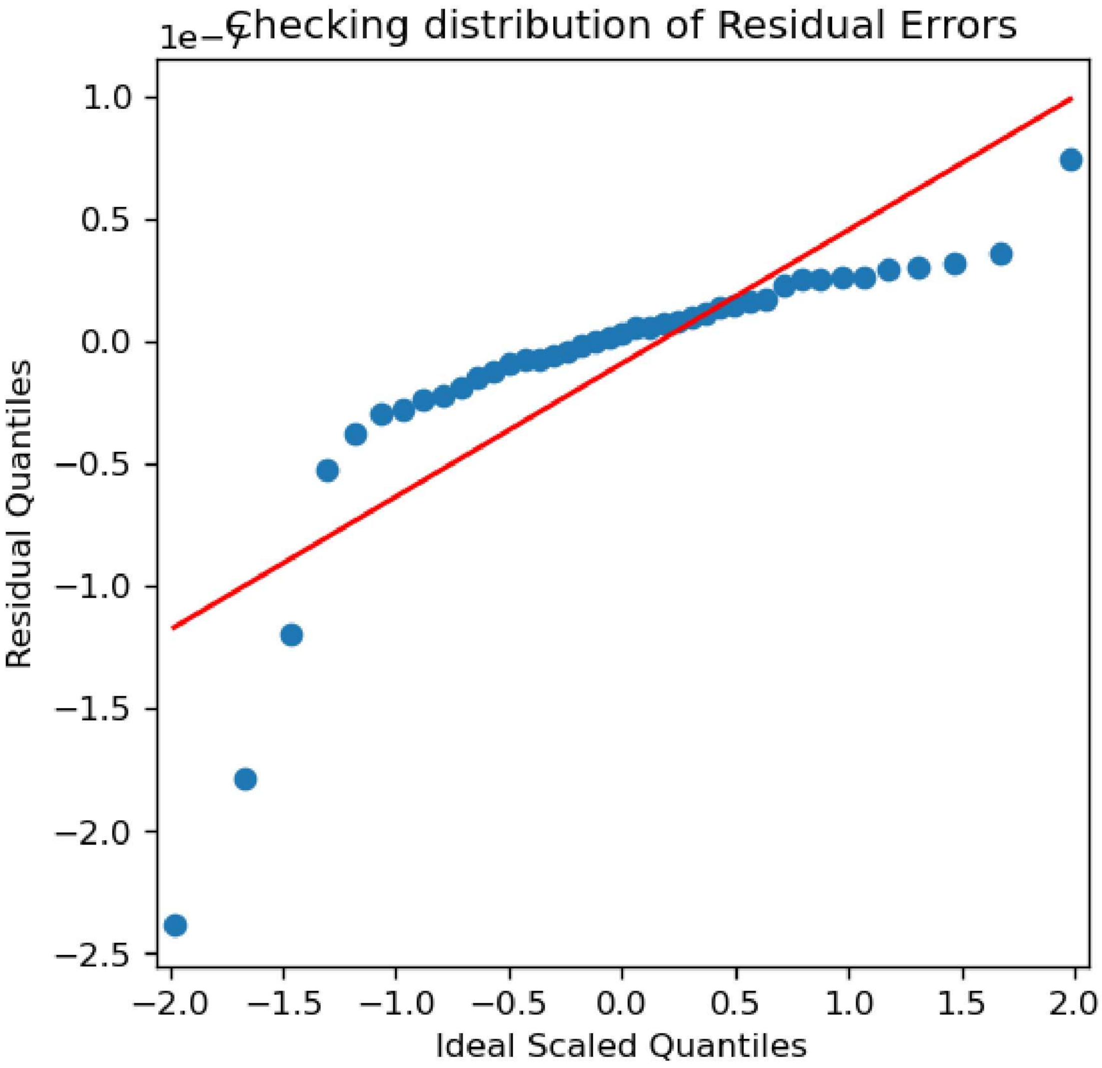
Checking Distribution of Residuals

In [18]:
```
plt.figure(figsize=(10, 6), dpi=120, facecolor='w', edgecolor='b')
plt.hist(residuals.residuals, bins = 150)
plt.xlabel('Error')
plt.ylabel('Frequency')
plt.title('Distribution of Error Terms')
plt.show()
```



QQ-Plot

In [19]:
```
# importing the QQ-plot from the from the statsmodels
from statsmodels.graphics.gofplots import qqplot

## Plotting the QQ plot
fig, ax = plt.subplots(figsize=(5,5) , dpi = 120)
qqplot(residuals.residuals, line = 's' , ax = ax)
plt.ylabel('Residual Quantiles')
plt.xlabel('Ideal Scaled Quantiles')
plt.title('Checking distribution of Residual Errors')
plt.show()
```



Checking distribution of Residual Errors
1e−7
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Residual Quantiles
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
Ideal Scaled Quantiles


```
In [ ]:
```