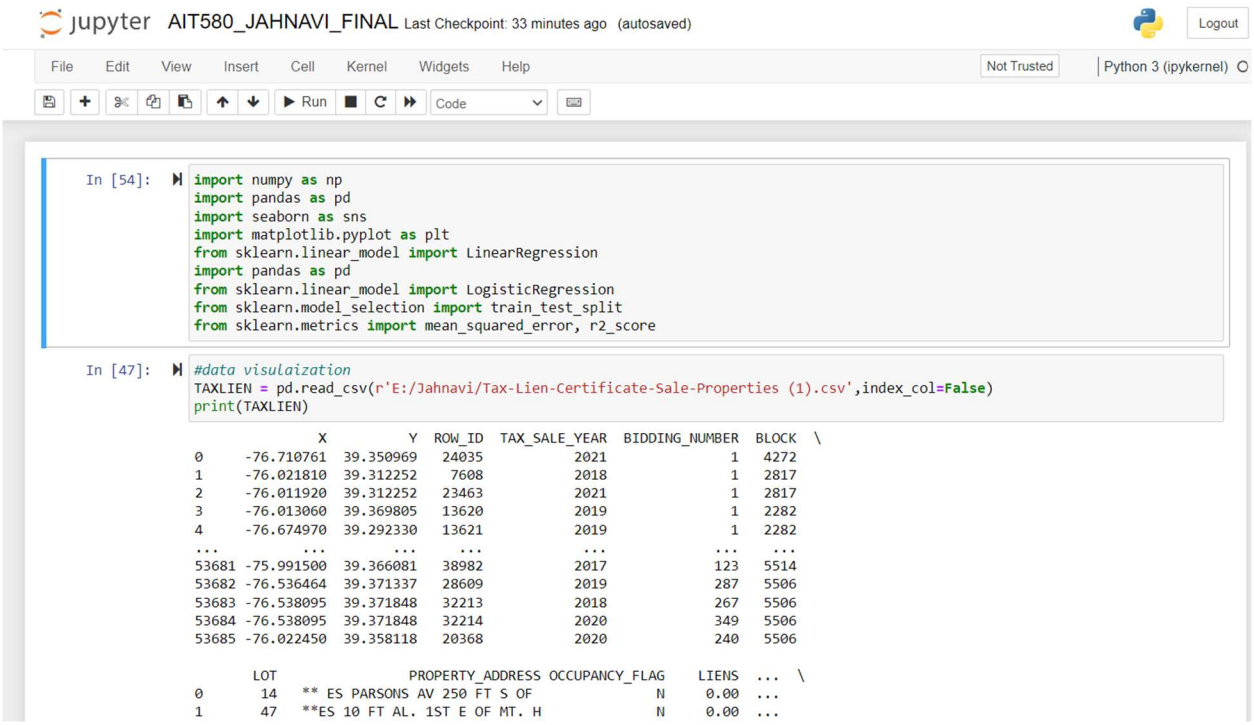
PYTHON VISUALIZATION AND REGRESSION

IMPORTING PACKAGE AND UPLOADING DATASET



FINDING THE COEFFICIEN:

The first two lines of the code select the variables for the regression analysis. 'X' represents the independent variable, which is the number of liens, and 'y' represents the dependent variable, which is the total amount of tax lien. These variables are extracted from a Pandas dataframe called 'TAXLIEN'.

Next, a linear regression model is fitted to the data using the 'LinearRegression' class from Scikit-Learn. The 'fit' method is called on the model object, passing in the independent variable 'X' and the dependent variable 'y'. This method fits the linear regression model to the data.

After fitting the model, the coefficients of the regression equation are printed to the console using the 'print' function. The 'intercept' represents the constant term in the regression equation, and the 'coef_' attribute represents the coefficient(s) of the independent variable(s). These values indicate the slope and the intercept of the regression line, respectively.

Overall, this code performs a simple linear regression analysis to estimate the relationship between the number of liens and the total amount of tax lien.

In [48]:

```
# select variables for the regression analysis
X = TAXLIEN[['LIENS']]
y = TAXLIEN['TOTAL_AMOUNT']

# fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# print the coefficients
print('Intercept:', model.intercept_)
print('Coefficients:', model.coef_)
```

```
Intercept: 201.98276619328135
Coefficients: [1.00008043]
```

FINDING THE LINEAR REGRESSION:

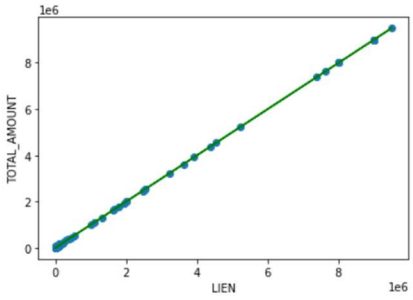
This code performs a simple linear regression analysis between two variables in a dataset and plots a scatter plot along with a regression line to visualize the relationship between the two variables.

The code first extracts two columns from a Pandas dataframe called 'TAXLIEN': 'LIENS' and 'TOTAL_AMOUNT' which are assigned to 'x' and 'y' respectively. 'x' represents the independent variable (i.e., the predictor variable), which is the number of liens, and 'y' represents the dependent variable (i.e., the response variable), which is the total amount of tax lien.

The next lines of code create a scatter plot of the data using Matplotlib's 'scatter()' function. Then, a green regression line is plotted using Matplotlib's 'plot()' function by passing in the independent variable 'x' and the predicted values of the dependent variable 'Y_pred'. The 'xlabel()' and 'ylabel()' functions are used to label the axes of the plot.

Finally, the plot is displayed using the 'show()' function of Matplotlib.

Overall, this code is a simple and effective way to visualize the relationship between two variables and estimate the regression line that best fits the data.



DOING CORRELATION MATRIX:

This code calculates and prints the correlation matrix for a Pandas dataframe called 'TAXLIEN'.

The 'corr()' method of a Pandas dataframe calculates the pairwise correlations between all the columns in the dataframe. The resulting correlation matrix is a square matrix where the diagonal elements are always 1 (i.e., the correlation between a variable and itself is always perfect).

The code calculates the correlation matrix by calling the 'corr()' method on the 'TAXLIEN' dataframe and assigns the resulting matrix to the variable 'corr_matrix'.

The next line of code prints the correlation matrix to the console using the 'print()' function. The correlation matrix is a tabular representation of the pairwise correlations between the variables in the dataframe, where each cell contains the correlation coefficient between two variables. The coefficient ranges from -1 to 1, where -1 represents a perfect negative correlation, 0 represents no correlation, and 1 represents a perfect positive correlation.

Overall, this code is useful for understanding the pairwise relationships between the variables in a dataset and identifying any strong correlations that may exist between the variables.

In [50]:

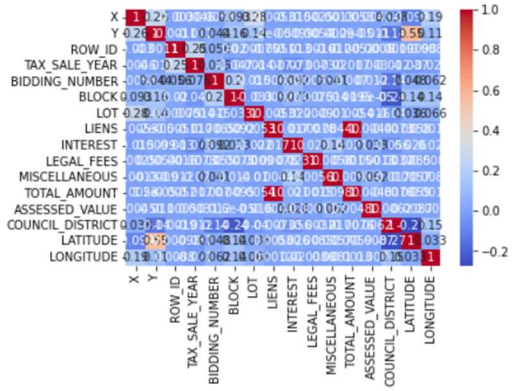
```
# Calculate the correlation matrix
corr_matrix = TAXLIEN.corr()

# Print the correlation matrix
print(corr_matrix)
```

```
                          X         Y    ROW_ID  TAX_SALE_YEAR  BIDDING_NUMBER  \
X                  1.000000  0.259440 -0.002994       0.004634        0.005844
Y                  0.259440  1.000000 -0.001075      -0.010749        0.043975
ROW_ID            -0.002994 -0.001075  1.000000       0.251288        0.055758
TAX_SALE_YEAR      0.004634 -0.010749  0.251288       1.000000        0.075528
BIDDING_NUMBER     0.005844  0.043975  0.055758       0.075528        1.000000
BLOCK              0.093232  0.159587  0.020020      -0.046959        0.196653
LOT                0.283336  0.135456 -0.000749       0.001420        0.014566
LIENS             -0.005254 -0.000020 -0.005078      -0.017005        0.006459
INTEREST          -0.014838  0.009877 -0.012772      -0.072762        0.092058
LEGAL_FEES         0.002538  0.005399 -0.001585      -0.007255        0.005490
MISCELLANEOUS     -0.001341 -0.000897 -0.011555      -0.020105        0.040599
TOTAL_AMOUNT      -0.005309  0.000012 -0.005178      -0.017351        0.006970
ASSESSED_VALUE     0.004458 -0.010722 -0.000502       0.002985        0.012315
COUNCIL_DISTRICT   0.038265 -0.139345 -0.001869      -0.011862       -0.142799
LATITUDE          -0.099246  0.550886 -0.000985      -0.037246        0.048404
LONGITUDE          0.189382  0.108392  0.008754      -0.021494        0.062241
```

PROVIDING THE HEAD MAP:

In [51]:

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



SELECT THE COLUMNS SHOWS HIGH ASSESSED_VALUE IN EACH BLOCK

This code selects the columns of a Pandas dataframe called 'TAXLIEN' that show high assessed values in each block, creates a scatter plot of the data, and labels the axes.

The code first selects two columns from the 'TAXLIEN' dataframe: 'BLOCK' and 'ASSESSED_VALUE'. 'X' represents the independent variable, which is the block, and 'y' represents the dependent variable, which is the assessed value of the properties in each block.

Finally, the plot is displayed using the 'show()' function of Matplotlib.

Overall, this code is useful for visualizing the relationship between the block and assessed value of properties in a dataset and identifying any trends or patterns that may exist in the data.

In [68]:

```
# Select the columns SHOWS HIGH ASSESSED_VALUE IN EACH BLOCK
X = TAXLIEN[['BLOCK']].values
y = TAXLIEN['ASSESSED_VALUE'].values


# Plot the results
plt.scatter(X[:, 0], y, color='blue')
plt.xlabel('BLOCK')
plt.ylabel('ASSESSED_VALUE')
plt.title('BLOCK vs ASSESSED_VALUE')
plt.show()
```