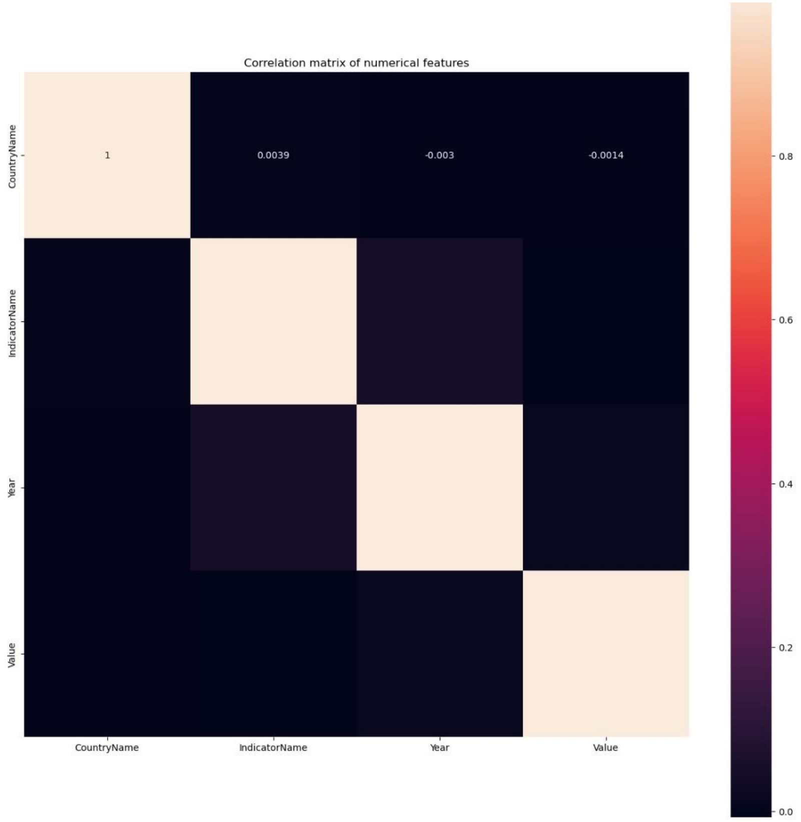
Data Collection and Preprocessing Phase

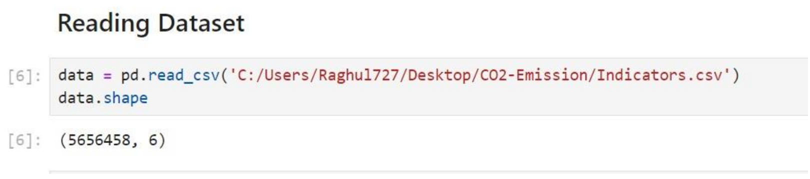
Date	15 March 2024
Team ID	SWTID1720007638
Project Title	Predicting Co2 Emission By Countries Using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

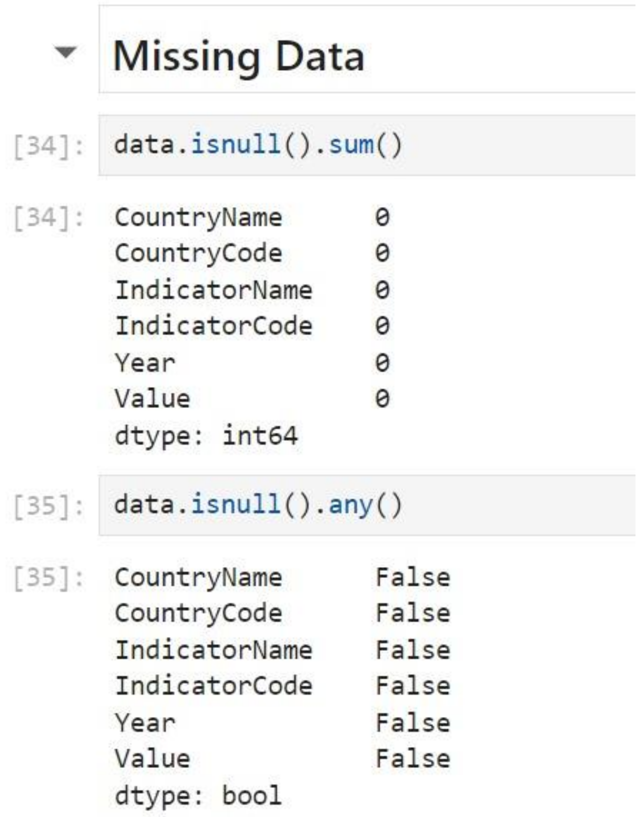
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	**Description**
Data Overview	1. Structure and Size: The dataset is a pandas Data Frame with 5,656,458 rows and 6 columns, using approximately 258.9+ MB of memory. 2.Columns and Types: It includes \`Country Name\`, \`Country Code\`, \`Indicator Name\`, \`Indicator Code\` (all objects), \`Year\` (int64), and \`Value\` (float64). 3. Year Statistics: The \`Year\` column ranges from 1960 to 2015, with a mean of 1994.464, a median of 1997, and a standard deviation of 13.87895. 4. Value Statistics: The \`Value\` column ranges from -9.82482e+15 to 1.103367e+16, with a mean of 1.070501e+12, a median of 63.57450, and a standard deviation of 4.842469e+13. 5. Additional Observations: The dataset includes various indicators for different countries over 55 years, with diverse value ranges and some negative values indicating specific metrics.
Univariate Analysis	Given below are the Univariate Analysis of "Year" and "Value": YEAR: Count: 5,656,458 Mean: 1,994.464 Standard Deviation: 13.879

	Minimum: 1,960 25th Percentile: 1,984 Median: 1,997 75th Percentile: 2,006 Maximum: 2,015 VALUE: Count: 5,656,458 Mean: 1,070,501,200 (1.07 billion) Standard Deviation: 48,424,690,000 (approximately 48.42 billion) Minimum: -9,824,821,500,000 (negative value indicates possible data issues) 25th Percentile: 5.566 Median: 63.575 75th Percentile: 13,467,220 Maximum: 11,033,670,000,000 (approximately 11 trillion)
Bivariate Analysis	 The above image displays a correlation matrix of numerical features, highlighting the weak correlations between "CountryName," "IndicatorName," "Year," and "Value”.
Multivariate Analysis	The correlation matrix illustrates the relationships among several numerical features, including "CountryName," "IndicatorName," "Year," and "Value." The values indicate

	weak correlations across all pairs: CountryName and IndicatorName: A negligible positive correlation of 0.0039 suggests that changes in country classifications have little influence on the indicators measured. CountryName and Year: A slight negative correlation of -0.003 indicates that as years progress, there is minimal variation in country classifications, reflecting stability over time. CountryName and Value: The correlation of -0.0014 shows virtually no relationship between country classifications and the measured values, indicating that country-specific attributes do not significantly affect the reported values. IndicatorName and Year: The correlation is not explicitly mentioned but is likely weak, similar to other pairs, suggesting that the indicators do not change drastically with time. IndicatorName and Value: The weak correlations imply that the indicators do not strongly predict the values, indicating a potential lack of direct causation between the two.
Data Preprocessing Code Screenshots	
Loading Data	Reading Dataset [6]: `data = pd.read_csv('C:/Users/Raghul727/Desktop/CO2-Emission/Indicators.csv')` `data.shape` [6]: `(5656458, 6)`

<table>
<tr>
<td>Handling Missing Data</td>
<td>



Missing Data

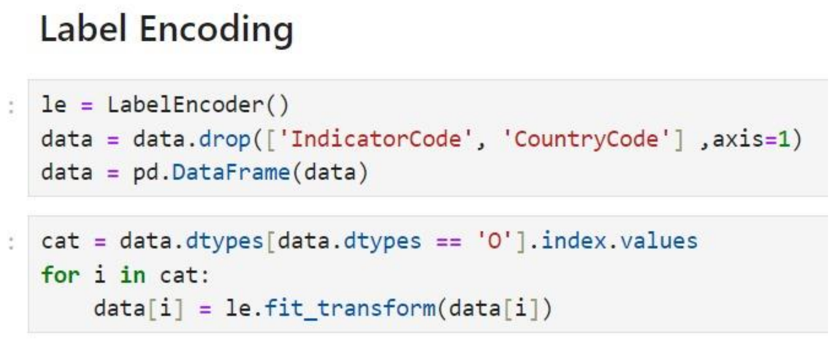
```
[34]: data.isnull().sum()

[34]: CountryName      0
      CountryCode      0
      IndicatorName    0
      IndicatorCode    0
      Year             0
      Value            0
      dtype: int64

[35]: data.isnull().any()

[35]: CountryName      False
      CountryCode      False
      IndicatorName    False
      IndicatorCode    False
      Year             False
      Value            False
      dtype: bool
```

</td>
</tr>
<tr>
<td>Data Transformation</td>
<td>



Label Encoding

```
le = LabelEncoder()
data = data.drop(['IndicatorCode', 'CountryCode'] ,axis=1)
data = pd.DataFrame(data)
```

```
cat = data.dtypes[data.dtypes == 'O'].index.values
for i in cat:
    data[i] = le.fit_transform(data[i])
```

</td>
</tr>
</table>