**import pandas as pd**

**import matplotlib.pyplot as plt**

**class DataAnalyzer:**

**def \_\_init\_\_(self, file\_path):**

**self.file\_path = file\_path**

**self.data = None**

**def load\_data(self):**

**try:**

**if self.file\_path.endswith('.csv'):**

**self.data = pd.read\_csv(self.file\_path)**

**elif self.file\_path.endswith('.xlsx'):**

**self.data = pd.read\_excel(self.file\_path)**

**else:**

**raise ValueError("Unsupported file format. Only CSV and XLSX are supported.")**

**except Exception as e:**

**print(f"Error: {e}")**

**def print\_numeric\_summary(self):**

**# Print summaries of all numeric variables in the dataset**

**if self.data is None:**

**print("No data loaded. Please load the data first using the 'load\_data' method.")**

**else:**

**numeric\_data = self.data.select\_dtypes(include='number')**

**print(numeric\_data.describe())**

**def generate\_bar\_graph(self, categorical\_variable):**

**# Generate a bar graph for a specified categorical variable**

**if self.data is None:**

**print("No data loaded. Please load the data first using the 'load\_data' method.")**

**else:**

**if categorical\_variable not in self.data.columns:**

**print(f"Error: '{categorical\_variable}' is not a valid categorical variable.")**

**else:**

**counts = self.data[categorical\_variable].value\_counts()**

**counts.plot(kind='bar')**

**plt.title(f"Bar Graph for '{categorical\_variable}'")**

**plt.xlabel(categorical\_variable)**

**plt.ylabel("Count")**

**plt.show()**

**def plot\_scatter\_plot(self, x\_var, y\_var):**

**# Plot scatter plots for two specified numeric variables**

**if self.data is None:**

**print("No data loaded. Please load the data first using the 'load\_data' method.")**

**else:**

**if x\_var not in self.data.columns or y\_var not in self.data.columns:**

**print(f"Error: '{x\_var}' or '{y\_var}' is not a valid numeric variable.")**

**else:**

**plt.scatter(self.data[x\_var], self.data[y\_var])**

**plt.title(f"Scatter Plot for '{x\_var}' vs '{y\_var}'")**

**plt.xlabel(x\_var)**

**plt.ylabel(y\_var)**

**plt.show()**