```
 1. Variables & File Handling
self.data_filename = data_filename
self.category_filename = category_filename
self.data = []
self.unique_categories = set()
self.results = {}
1a. Reading the numerical file
   with open(self.data_filename, 'r') as file:
      lines = file.readlines()
 1b. Storing data in variables (list)
     self.data.append(float(clean_line))
      except ValueError:


def load_data(self):
  """Reads numerical data and categorical data with error handling."""
```

2. Error Handling

```
if not os.path.exists(self.data_filename):
    print(f"Error: The file '{self.data_filename}' does not exist.")
    return False
```

```
    2b. Handle empty file
    if not lines:
        print("Error: The data file is empty.")
        return False

    for line in lines:
        clean_line = line.strip()
        if clean_line:
            try:
2b. Handle non-numeric values
            print(f"Skipping invalid non-numeric value: {clean_line}")
```

3. Functions & 4. Operators & Loops

```
        def calculate_statistics(self):
    """Calculates stats using manual loops and arithmetic operators."""
     if not self.data:
      return



 4. Initialize variables for manual calculation
total = 0
count = 0
minimum = self.data[0]
maximum = self.data[0]

4a. Arithmetic operator (/) for average
average = total / count

 Store results in a dictionary
  self.results = {
   "Total": total,
   "Average": average,
   "Minimum": minimum,
   "Maximum": maximum,
   "Count": count
}
```

```
        4b. Loop to traverse the dataset
        for value in self.data:
            total += value  # 4a. Arithmetic operator (+)
            count += 1


            4b. Compare for min and max
            if value < minimum:
                minimum = value
            if value > maximum:
                maximum = value



    def display_results(self, threshold=75):
        """Displays data and evaluates performance."""



5. Conditional Statements
        if self.results["Average"] > threshold:
            print("Performance Status: High Performance")
        else:
            print("Performance Status: Needs Improvement")
```

```
            6. Sets - Extracting unique categories
            if os.path.exists(self.category_filename):
                with open(self.category_filename, 'r') as cat_file:
                    6a. Using a set for unique values
                    self.unique_categories = {line.strip() for line in cat_file if line.strip()}

                return True
            except Exception as e:
                print(f"An unexpected error occurred: {e}")
                return False

        6b. Display unique categories
            print(f"Total Unique Categories: {len(self.unique_categories)}")
            print(f"Categories: {', '.join(self.unique_categories)}")

        def save_results(self, output_filename="report.txt"):

7. Object-Oriented Programming (OOP)
class DataSet:
    def __init__(self, data_filename, category_filename):
        7a. Method to display results
        if not self.results:
            print("No statistics available.")
            return
```

```
        print("\n--- Statistics Report ---")
        for key, value in self.results.items():
            print(f"{key}: {value:.2f}")
```

7b. Create an object and run the analysis

```
if __name__ == "__main__":
    Example usage:
    Ensure 'data.txt' and 'categories.txt' exist in your folder
    analyzer = DataSet("data.txt", "categories.txt")

    if analyzer.load_data():
        analyzer.calculate_statistics()
        analyzer.display_results(threshold=60) # Customize threshold here
        analyzer.save_results()
```

```
        """8. Saving Results to a report file."""
        try:
            with open(output_filename, 'w') as f:
                f.write("DATA ANALYSIS SUMMARY\n")
                f.write("--------------------\n")
```

```
            for key, value in self.results.items():
                f.write(f"{key}: {value:.2f}\n")
            f.write(f"\nUnique Categories ({len(self.unique_categories)}): ")
            f.write(", ".join(self.unique_categories))
        print(f"\n[Success] Report saved to {output_filename}")
    except IOError:
        print("Error: Could not save the report file.")
```