```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', None)


class DataSet:
    def __init__(self, df):
        self.df = df

    def compute_statistical_summary(self):
        numerical_columns = self.df.select_dtypes(include=[np.number])
        if not numerical_columns.empty:
            numerical_stats = numerical_columns.agg(['mean', 'median', 'std'])
            print("Statistics for numerical columns:")
            print(numerical_stats)
        else:
            print("No numerical columns found.")

        categorical_columns = self.df.select_dtypes(include=['object', 'category'])
        if not categorical_columns.empty:
            print("\nFrequency distribution for categorical columns:")
            for column in categorical_columns:
                freq_dist = ((self.df[column].value_counts() / len(df)) * 100).astype(str) + ' %'
                print(f"\nColumn: {column}")
                print(freq_dist)

    def visualize_data(self):
        sns.set_palette('bright')
```

```
36          font_title = {'fontweight': 'bold', '
   fontsize': 14}
37          font_label = {'fontweight': 'bold', '
   fontsize': 12}
38          numerical_columns = self.df.select_dtypes(
   include=[np.number])
39          for column in numerical_columns:
40              plt.figure()
41              sns.histplot(self.df[column])
42              plt.title(f"Histogram of {column}", **
   font_title)
43              plt.xlabel(column, **font_label)
44              plt.ylabel("Frequency", **font_label)
45              plt.show()
46          for column in numerical_columns:
47              plt.figure()
48              sns.boxplot(x=self.df[column])
49              plt.title(f"Box plot of {column}", **
   font_title)
50              plt.xlabel(column, **font_label)
51              plt.ylabel("Value", **font_label)
52              median_value = df[column].median()
53              plt.text(0.5, 0.5, f"Median: {
   median_value:.2f}", transform=plt.gca().transAxes,
54                       horizontalalignment='center',
   verticalalignment='center', **font_label)
55              plt.show()
56          sns.set(font_scale=1.2)
57          sns.pairplot(self.df[numerical_columns.
   columns])
58          plt.title("Scatter Plot Matrix", **
   font_title)
59          plt.show()
60          categorical_columns = df.select_dtypes(
   include=['object', 'category'])
61          for column in categorical_columns:
62              plt.figure()
63              sns.countplot(data=df, x=column)
64              plt.title(f"Bar Graph of {column}", **
   font_title)
65              plt.xlabel(column, **font_label)
```

```
66              plt.ylabel("Count", **font_label)
67              value_counts = df[column].value_counts
    ()
68              for i, count in enumerate(value_counts
    ):
69                  plt.text(i, count, str(count), ha=
    'center', va='bottom', **font_label)
70              plt.show()
71
72      def preprocess_data(self):
73          numerical_columns = self.df.select_dtypes(
    include=[np.number])
74          scaler = MinMaxScaler()
75          self.df[numerical_columns.columns] =
    scaler.fit_transform(self.df[numerical_columns.
    columns])
76          categorical_columns = self.df.
    select_dtypes(include=['object', 'category']).
    columns.tolist()
77          self.df = pd.get_dummies(self.df, columns=
    categorical_columns)
78
79      def find_optimal_clusters(self):
80          X = self.df.drop('Customer_Id', axis=1)
81          silhouette_scores = []
82          k_range = range(2, 21)
83          max_k = 2
84          max_score = -1
85          for k in k_range:
86              print(f"Running for k = {k}")
87              kmeans = KMeans(n_clusters=k)
88              kmeans.fit(X)
89              labels = kmeans.predict(X)
90              score = silhouette_score(X, labels)
91              silhouette_scores.append(score)
92              if score > max_score:
93                  max_k = k
94                  max_score = score
95          print("Silhouette scores: ",
    silhouette_scores)
96          return max_k
```

```
 97
 98     def perform_cluster_and_analyze(self,
    optimal_clusters):
 99         X = self.df.drop('Customer_Id', axis=1)
100         kmeans = KMeans(n_clusters=
    optimal_clusters)
101         kmeans.fit(X)
102         print("Number of customers in each cluster
    :", kmeans.labels_.size)
103         cluster_labels = kmeans.labels_
104         unique_labels, counts = np.unique(
    cluster_labels, return_counts=True)
105         for cluster_label, count in zip(
    unique_labels, counts):
106             print(f"Cluster {cluster_label}: {
    count} samples")
107         X['Cluster'] = cluster_labels
108         cluster_means = X.groupby('Cluster').mean
    ()
109         print(cluster_means)
110         cluster_means.to_csv('cluster_means.csv')
111
112
113 if __name__ == '__main__':
114     df = pd.read_csv("JustBuy_data.csv")
115     data_set = DataSet(df)
116     data_set.compute_statistical_summary()
117     data_set.visualize_data()
118     data_set.preprocess_data()
119     optimal_clusters = data_set.
    find_optimal_clusters()
120     print("Optimal number of clusters: ",
    optimal_clusters)
121     data_set.perform_cluster_and_analyze(
    optimal_clusters)
122
123
```