```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

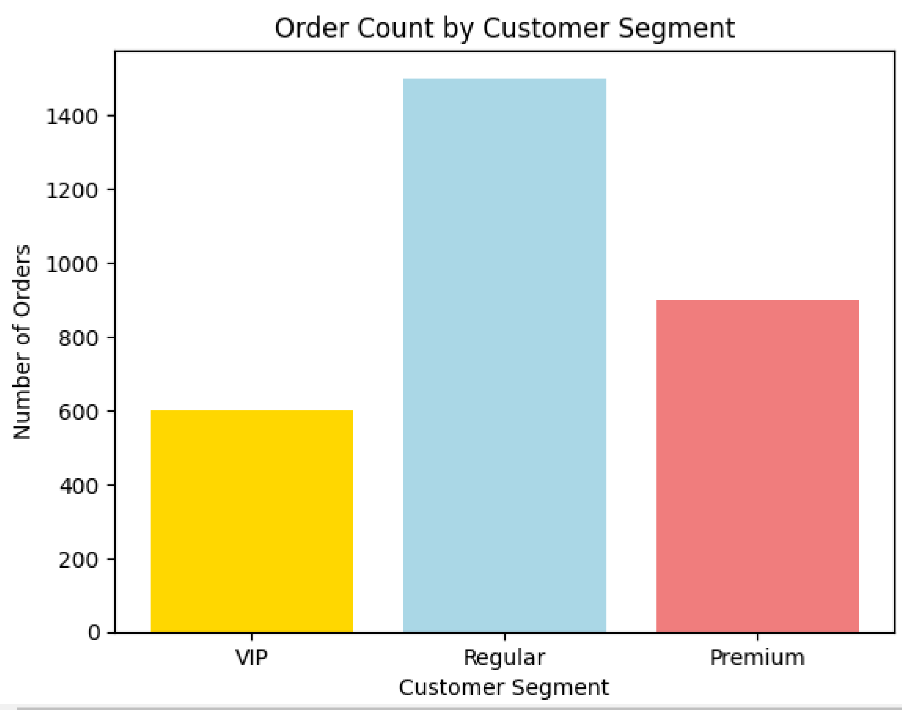
```
# Load the dataset from a CSV file
data = pd.read_csv('/content/Order_Data.csv')
customer_data = pd.read_csv('/content/Customer_Data.csv')

# Display the first few rows of the dataset
print(customer_data.columns)
print(customer_data.head())
```

```
Index(['Customer_ID', 'First_Name', 'Last_Name', 'Email', 'Phone', 'Gender',
       'Birthdate', 'Join_Date', 'Location', 'Segment', 'Date_Assigned'],
      dtype='object')
  Customer_ID First_Name  Last_Name                              Email  \
0        1.00       Erin      Brown               ianjordan@morales.net
1        2.00        Ana   Thompson  christopheribarra@brooks-owens.com
2        3.00      Julie     Norman     petertaylor@williams-baxter.org
3        4.00     Thomas      Baker               gbowen@kent-cruz.com
4        5.00      Emily  Robertson            johnreyes@zimmerman.com

                   Phone  Gender   Birthdate   Join_Date         Location  \
0      (442)749-3583x632    Male  12/26/2000    8/5/2022     East Michael
1      026-722-3158x9970    Male   7/23/1967    7/6/2024  New Patriciaton
2      (613)840-4935x691    Male   4/18/2001   3/13/2021         Maryport
3     960-566-7967x85572  Female   3/19/1977  12/28/2020  Port Thomasport
4  001-106-300-1836x23763    Male   2/16/1984   1/16/2020   Port Monicaton

   Segment Date_Assigned
0  Regular      6/3/2021
1      VIP     12/2/2022
2  Premium     10/3/2023
3      VIP     12/1/2022
4  Premium     8/23/2021
```

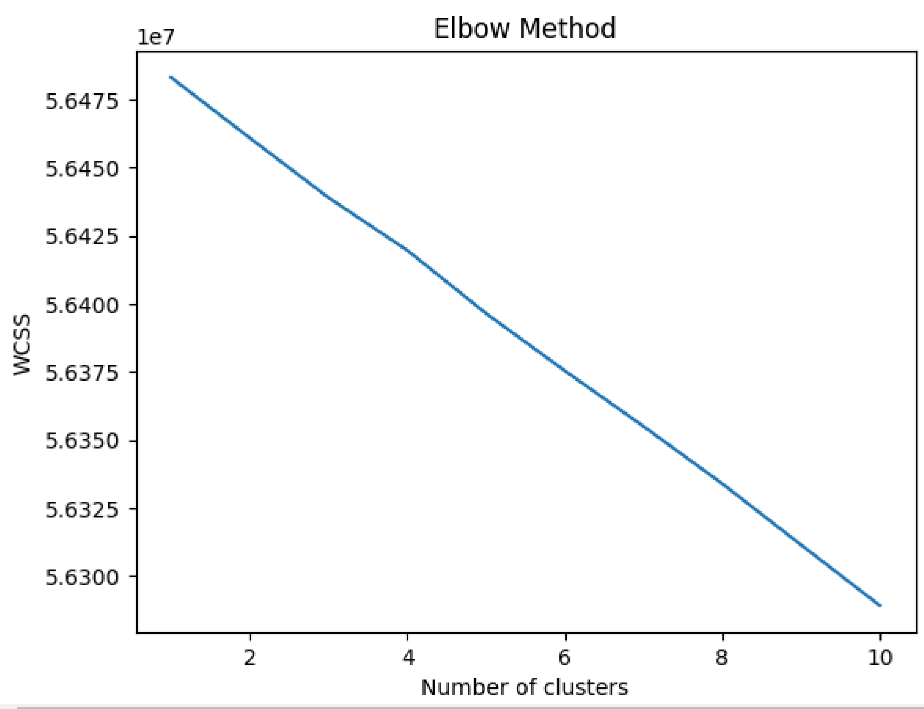
```
# Assuming 'customer_data' contains the data you want to analyze
# If you have a different DataFrame, replace 'customer_data' with its name
merged_data_fixed = customer_data

# Group by the 'Segment' column and count the number of orders for each segment
order_count_by_segment = merged_data_fixed.groupby('Segment').size()

# Output the count for each segment
vip_orders = order_count_by_segment.get('VIP', 0)
regular_orders = order_count_by_segment.get('Regular', 0)
premium_orders = order_count_by_segment.get('Premium', 0)

# Display the results
print(f"VIP Orders: {vip_orders}")
print(f"Regular Orders: {regular_orders}")
print(f"Premium Orders: {premium_orders}")
```

```
VIP Orders: 599
Regular Orders: 1501
Premium Orders: 899
```

```
# Assuming 'customer_data' contains the data you want to analyze
# If you have a different DataFrame, replace 'customer_data' with its name
merged_data_fixed = customer_data

# Group by the 'Segment' column and count the number of orders for each segment
order_count_by_segment = merged_data_fixed.groupby('Segment').size()

# Output the count for each segment
vip_orders = order_count_by_segment.get('VIP', 0)
regular_orders = order_count_by_segment.get('Regular', 0)
premium_orders = order_count_by_segment.get('Premium', 0)

# Create a bar chart
segments = ['VIP', 'Regular', 'Premium']
order_counts = [vip_orders, regular_orders, premium_orders]

plt.bar(segments, order_counts, color=['gold', 'lightblue', 'lightcoral'])
plt.xlabel("Customer Segment")
plt.ylabel("Number of Orders")
plt.title("Order Count by Customer Segment")
plt.show()
```



```
# Data Preprocessing
df = customer_data
df = pd.get_dummies(df, drop_first=True)
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df)
#build predective model
# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    # Indent the following lines within the for loop
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(scaled_df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
# Fit the model with the optimal number of clusters
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=42)
y_kmeans = kmeans.fit_predict(scaled_df)
# Add the cluster labels to the original dataframe
df['Cluster'] = y_kmeans
```

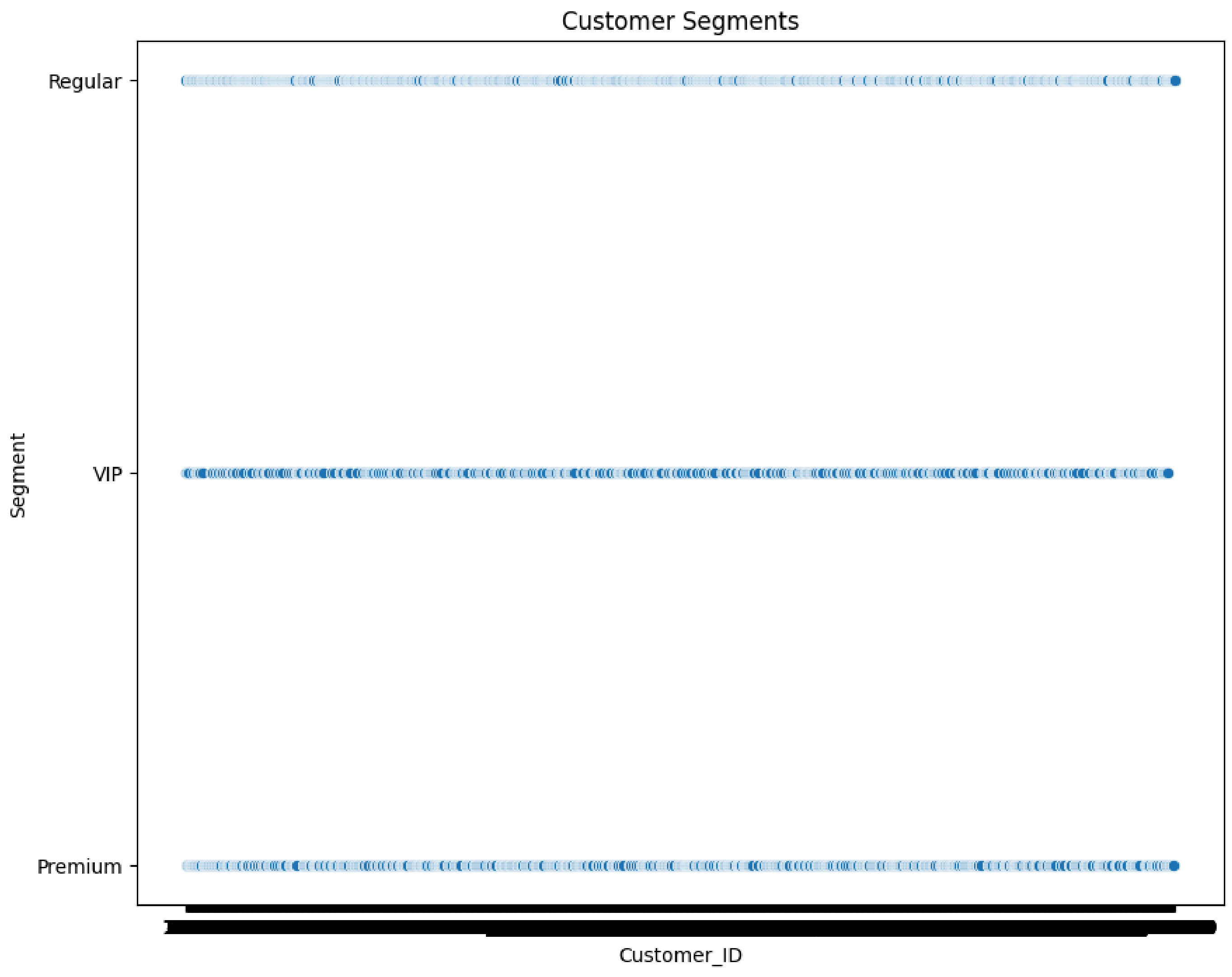


```
#evaluate the model
score = silhouette_score(scaled_df, y_kmeans)
print(f'Silhouette Score: {score}')
```

```
Silhouette Score: 0.016553613833014097
```

```
#visualize model
plt.figure(figsize=(10, 8))
df = customer_data
# Check if 'Customer_ID' exists in the DataFrame columns
if 'Customer_ID' in customer_data.columns:
    sns.scatterplot(x=df['Customer_ID'], y=df['Segment'], palette='viridis')
else:
    # If 'Customer_ID' is not found, print a message or use a different column
    print("Column 'Customer_ID' not found in the DataFrame.")
    # Optionally, use a different column for the x-axis
    # sns.scatterplot(x=df['another_column'], y=df['Segment'], palette='viridis')
plt.title('Customer Segments')
plt.show()
```

```
<ipython-input-98-34b7f4800082>:6: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(x=df['Customer_ID'], y=df['Segment'], palette='viridis')
```


Customer Segments
Regular
VIP
Premium
Segment
Customer_ID