
CUSTOMER SUSTOMERS
HIGH-VALUE CUSTOMERS
BUDGET CUSTOMERS
25%
FEQUENT BUYERS
FEQUENT SHODDERS
FEQUENT

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Data preprocessing (cleaning, handling missing values, etc.)

In [74]:
```
df=pd.read_csv("customer_segmentation_data.csv")
```

In [75]:
```
df.head()
```

Out[75]:

	Customer ID	Age	Gender	Marital Status	Education Level	Geographic Information	Occupation	Income Level	Behavioral Data	Purchase History	Interactions with Customer Service	Insurance Products Owned	Cover Amo
0	84966	23	Female	Married	Associate Degree	Mizoram	Entrepreneur	70541	policy5	04-10-2018	Phone	policy2	366
1	95568	26	Male	Widowed	Doctorate	Goa	Manager	54168	policy5	11-06-2018	Chat	policy1	780
2	10544	29	Female	Single	Associate Degree	Rajasthan	Entrepreneur	73899	policy5	06-05-2021	Email	policy3	773
3	77033	20	Male	Divorced	Bachelor's Degree	Sikkim	Entrepreneur	63381	policy5	09-02-2018	Chat	policy2	787
4	88160	25	Female	Separated	Bachelor's Degree	West Bengal	Manager	38794	policy1	09-10-2018	Chat	policy4	366

In [76]:
```
df.tail()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53503 entries, 0 to 53502
Data columns (total 20 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Customer ID                           53503 non-null  int64
 1   Age                                   53503 non-null  int64
 2   Gender                                53503 non-null  object
 3   Marital Status                        53503 non-null  object
 4   Education Level                       53503 non-null  object
 5   Geographic Information                53503 non-null  object
 6   Occupation                            53503 non-null  object
 7   Income Level                          53503 non-null  int64
 8   Behavioral Data                       53503 non-null  object
 9   Purchase History                      53503 non-null  object
 10  Interactions with Customer Service  53503 non-null  object
 11  Insurance Products Owned              53503 non-null  object
 12  Coverage Amount                       53503 non-null  int64
 13  Premium Amount                        53503 non-null  int64
 14  Policy Type                           53503 non-null  object
 15  Customer Preferences                  53503 non-null  object
 16  Preferred Communication Channel     53503 non-null  object
 17  Preferred Contact Time                53503 non-null  object
 18  Preferred Language                    53503 non-null  object
 19  Segmentation Group                    53503 non-null  object
dtypes: int64(5), object(15)
memory usage: 8.2+ MB
```

In [78]:
```
df.isnull().sum()
```

Out[79]:
```
Customer ID                          int64
Age                                  int64
Gender                              object
Marital Status                      object
Education Level                     object
Geographic Information              object
Occupation                          object
Income Level                         int64
Behavioral Data                     object
Purchase History                    object
Interactions with Customer Service  object
Insurance Products Owned            object
Coverage Amount                      int64
Premium Amount                       int64
Policy Type                         object
Customer Preferences                object
Preferred Communication Channel     object
Preferred Contact Time              object
Preferred Language                  object
Segmentation Group                  object
dtype: object
```

In [80]:
```
df.columns
```

Out[80]:
```
Index(['Customer ID', 'Age', 'Gender', 'Marital Status', 'Education Level',
       'Geographic Information', 'Occupation', 'Income Level',
       'Behavioral Data', 'Purchase History',
       'Interactions with Customer Service', 'Insurance Products Owned',
       'Coverage Amount', 'Premium Amount', 'Policy Type',
       'Customer Preferences', 'Preferred Communication Channel',
       'Preferred Contact Time', 'Preferred Language', 'Segmentation Group'],
      dtype='object')
```

In [81]:
```
df.describe().T
```

In [83]:
```
# Encoding categorical features
label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])
```

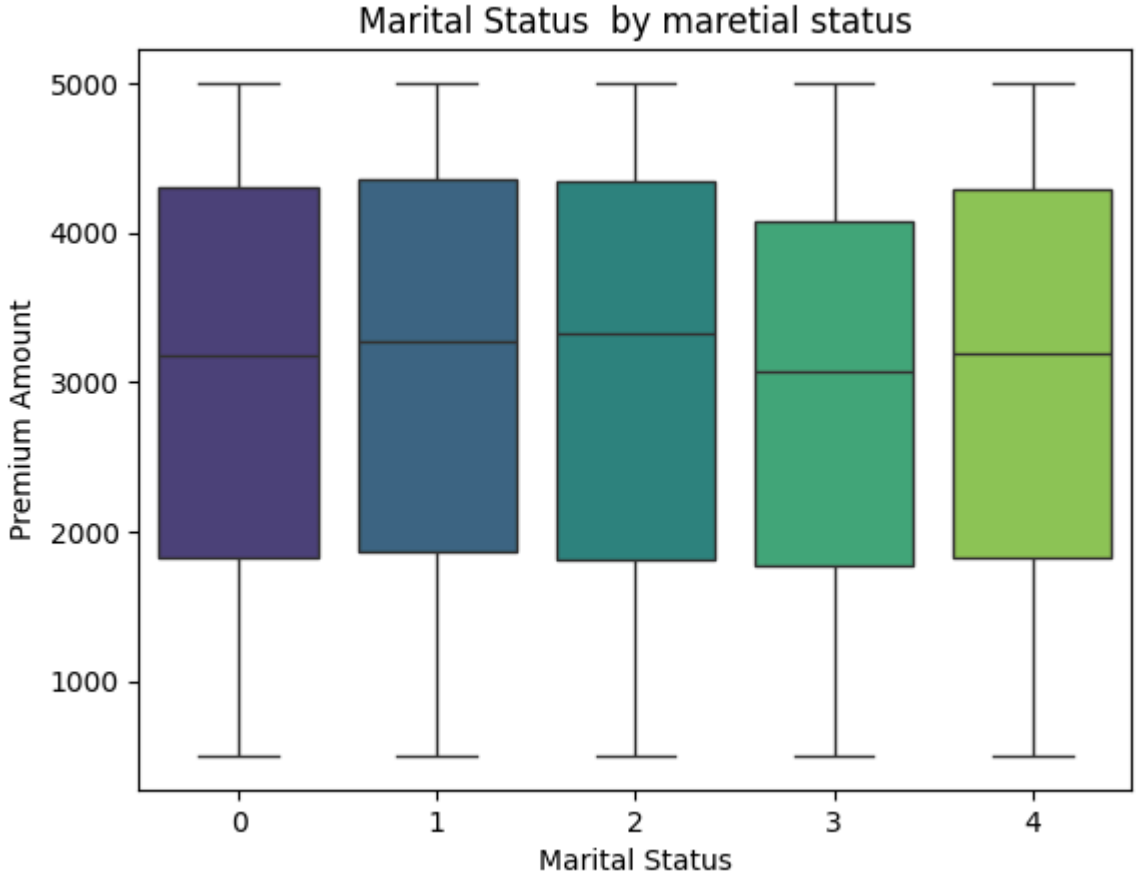
Customer Segmentation & Group Behavior

In [84]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="Marital Status",y="Premium Amount",data=df)
plt.xticks(rotation=45)
plt.title("Income lavel distributions Across segmentation group")
plt.show()
```

```
C:\Users\hi\AppData\Local\Temp\ipykernel_34588\301924940.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se
t `legend=False` for the same effect.

  sns.boxplot(x="Marital Status", y="Premium Amount", data=df,palette="viridis")
```



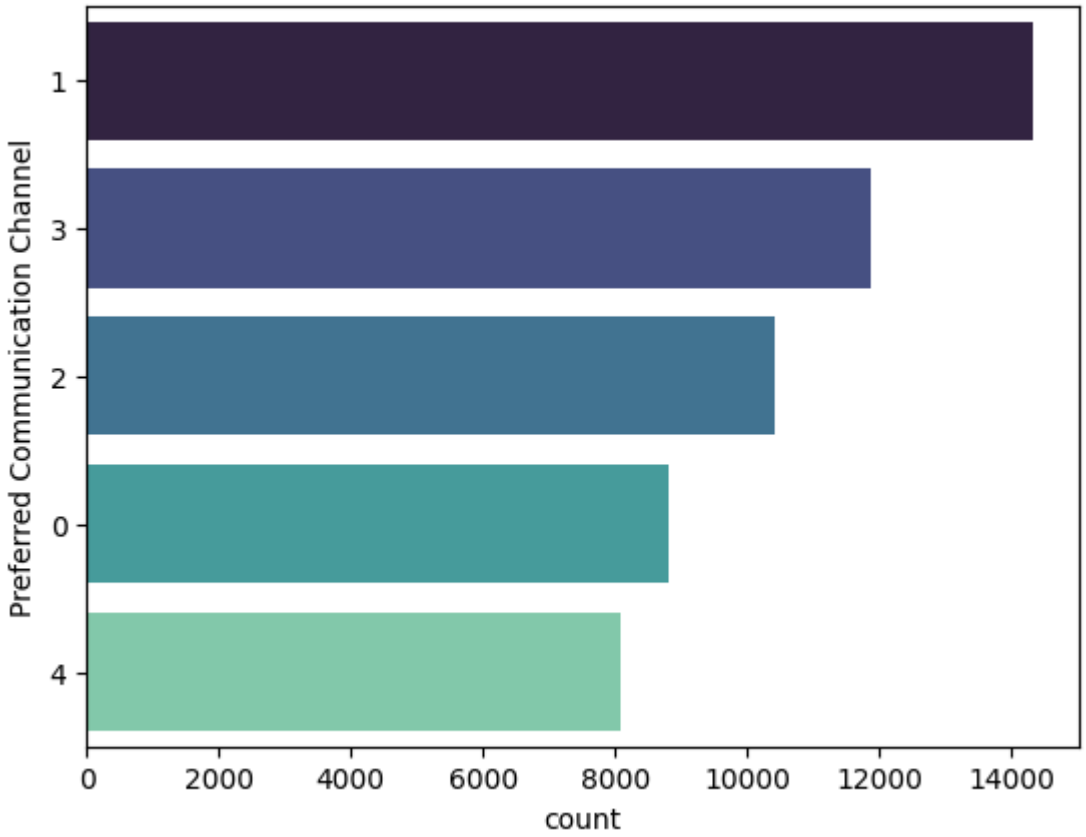
In [86]:
```
# Income Level vs. Segmentation Group
plt.figure(figsize=(10, 5))
sns.boxplot(x='Segmentation Group', y='Income Level', data=df, palette="Set2")
plt.xticks(rotation=45)
plt.title("Income Level Distribution Across Segmentation Groups")
plt.show()
```

```
C:\Users\hi\AppData\Local\Temp\ipykernel_34588\806573206.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and se
t `legend=False` for the same effect.

  sns.countplot(y="Preferred Communication Channel",data=df, order=df['Preferred Communication Channel'].value_counts().index,p
alette="mako")
```

Out[87]: <Axes: xlabel='count', ylabel='Preferred Communication Channel'>



In [88]:
```python
# Spending Trends by Age Group
plt.figure(figsize=(10, 5))
sns.scatterplot(x='Age', y='Premium Amount', hue='Segmentation Group', data=df, palette='tab10')
plt.title("Premium Amount Distribution by Age and Segmentation Group")
plt.show()
```

Step 2: Apply K-Means Clustering

In [90]:
```
# Select relevant numerical features
features = ['Age', 'Income Level', 'Premium Amount', 'Coverage Amount']
df_cluster = df[features]
```

In [91]:
```
# Standardize the features for better clustering
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cluster)
```

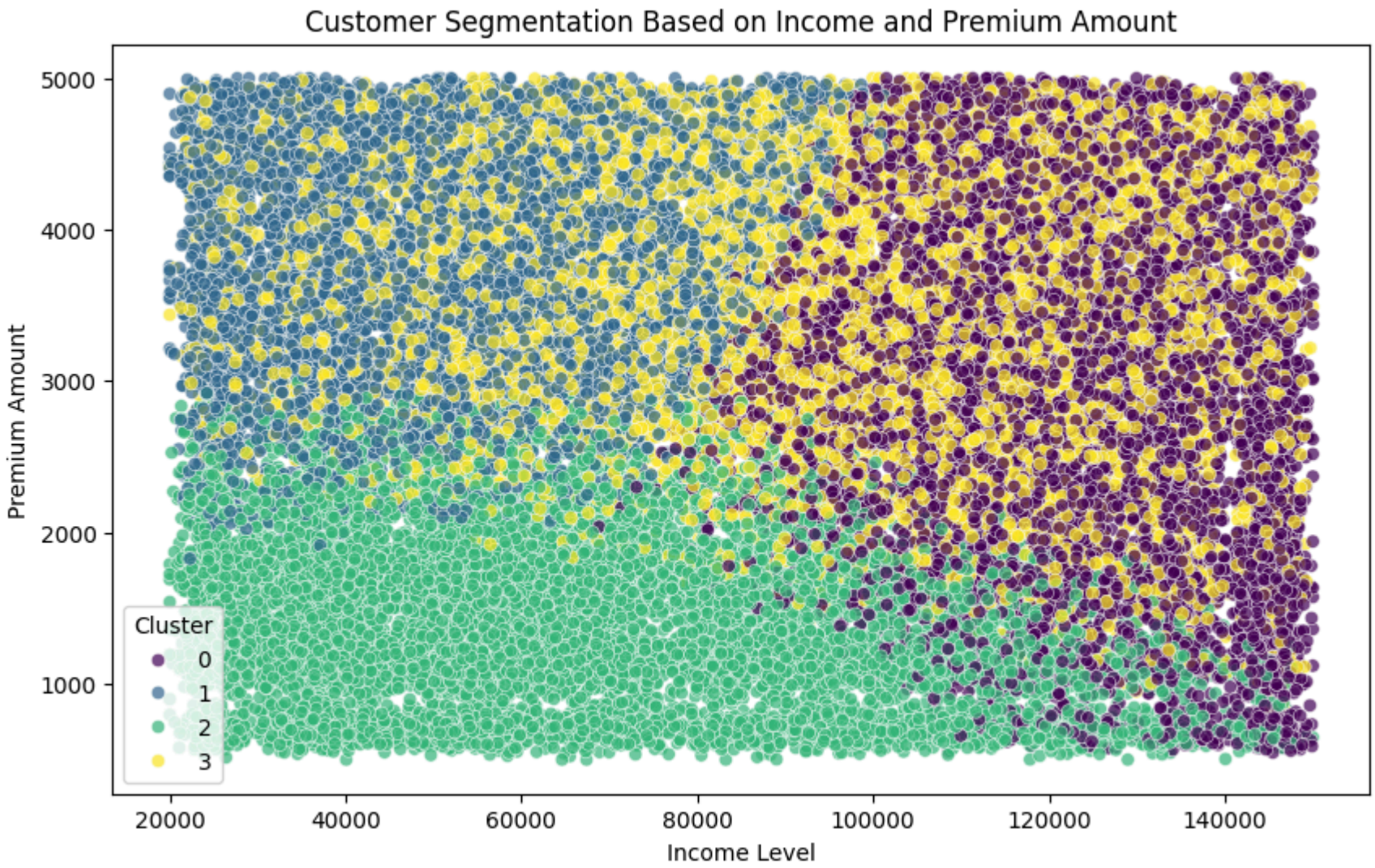
Determine the optimal number of clusters using the Elbow Method

In [92]:
```
inertia = []
K_range = range(1, 11)  # Testing for 1 to 10 clusters
```

In [93]:
```
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)
```

In [94]:
```
# Plot the Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia, marker='o', linestyle='--', color='b')
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia (WCSS)")
plt.title("Elbow Method to Find Optimal K")
plt.show()
```

```
plt.legend(title="Cluster")
plt.show()
```



In [97]:
```
# Display sample cluster data
print("\nSample Segmented Data:")
print(df[['Customer ID', 'Age', 'Income Level', 'Premium Amount', 'Coverage Amount', 'Cluster']].head(10))
```