```
import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 2025-VeloCityX-Expanded-Fan-Engagement-Data.csv to 2025-VeloCityX-Expanded-Fan-Engagement-Data (2).csv
```

```
filename = list(uploaded.keys())[0]
```

```
data= pd.read_csv(filename)
```

```
data.head()
```

	User ID	Fan Challenges Completed	Predictive Accuracy (%)	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)	Time on Live 360 (mins)	Real-Time Chat Activity (Messages Sent)
0	U001	5	80	3	10	120	20
1	U002	8	60	1	8	100	35
2	U003	3	90	0	6	90	5
3	U004	7	70	2	15	140	40
4	U005	2	50	5	3	60	8

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype 
---  ------                                  --------------  ----- 
 0   User ID                                 100 non-null    object
 1   Fan Challenges Completed                100 non-null    int64 
 2   Predictive Accuracy (%)                 100 non-null    int64 
 3   Virtual Merchandise Purchases           100 non-null    int64 
 4   Sponsorship Interactions (Ad Clicks)    100 non-null    int64 
 5   Time on Live 360 (mins)                 100 non-null    int64 
 6   Real-Time Chat Activity (Messages Sent)  100 non-null    int64 
dtypes: int64(6), object(1)
memory usage: 5.6+ KB
```

```
data.describe()
```

	Fan Challenges Completed	Predictive Accuracy (%)	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)	Time on Live 360 (mins)	Real-Time Chat Activity (Messages Sent)
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	5.790000	74.990000	2.670000	8.680000	129.350000	25.050000
std	2.825908	14.033506	2.064882	6.340315	38.634358	14.163101
min	1.000000	50.000000	0.000000	0.000000	60.000000	0.000000
25%	3.000000	62.000000	1.000000	2.000000	98.000000	11.000000
50%	6.000000	77.000000	2.000000	8.000000	124.500000	25.500000
75%	8.000000	86.500000	5.000000	15.000000	160.000000	35.000000
max	10.000000	98.000000	6.000000	19.000000	199.000000	49.000000

```
data.isnull().sum()
```

```
                                              0
User ID                                       0
Fan Challenges Completed                      0
Predictive Accuracy (%)                       0
Virtual Merchandise Purchases                 0
Sponsorship Interactions (Ad Clicks)          0
Time on Live 360 (mins)                       0
Real-Time Chat Activity (Messages Sent)       0

dtype: int64
```

```
duplicates= data.duplicated()
```

```
duplicates
```

```
          0
 0    False
 1    False
 2    False
 3    False
 4    False
...     ...
95    False
96    False
97    False
98    False
99    False
100 rows × 1 columns

dtype: bool
```

Observation: No missing value and No duplicate value.

```
data.dtypes
```

```
                                              0
User ID                                  object
Fan Challenges Completed                  int64
Predictive Accuracy (%)                   int64
Virtual Merchandise Purchases             int64
Sponsorship Interactions (Ad Clicks)      int64
Time on Live 360 (mins)                   int64
Real-Time Chat Activity (Messages Sent)   int64

dtype: object
```

```
data_new= data.drop(columns=['User ID'])
```

```
data_new
```

	Fan Challenges Completed	Predictive Accuracy (%)	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)	Time on Live 360 (mins)	Real-Time Chat Activity (Messages Sent)
0	5	80	3	10	120	20
1	8	60	1	8	100	35
2	3	90	0	6	90	5
3	7	70	2	15	140	40
4	2	50	5	3	60	8
...	...	...	...	...	...	...
95	8	86	6	14	98	22
96	3	82	3	1	159	24
97	1	91	1	9	92	34
98	8	93	0	1	160	40
99	3	73	3	16	82	29

100 rows × 6 columns

```
data_new.dtypes
```

	0
Fan Challenges Completed	int64
Predictive Accuracy (%)	int64
Virtual Merchandise Purchases	int64
Sponsorship Interactions (Ad Clicks)	int64
Time on Live 360 (mins)	int64
Real-Time Chat Activity (Messages Sent)	int64

dtype: object

```
data_new.fillna(data_new.median(), inplace=True)
```

```
columns_to_check = ['Fan Challenges Completed', 'Predictive Accuracy (%)',
                    'Virtual Merchandise Purchases', 'Sponsorship Interactions (Ad Clicks)',
                    'Time on Live 360 (mins)', 'Real-Time Chat Activity (Messages Sent)']
```

```
assert (data_new[columns_to_check] >= 0).all().all()
```

Observation: No negative values

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
data_new[columns_to_check] = scaler.fit_transform(data_new[columns_to_check])
```

```
data_new
```

	Fan Challenges Completed	Predictive Accuracy (%)	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)	Time on Live 360 (mins)	Real-Time Chat Activity (Messages Sent)
0	0.444444	0.625000	0.500000	0.526316	0.431655	0.408163
1	0.777778	0.208333	0.166667	0.421053	0.287770	0.714286
2	0.222222	0.833333	0.000000	0.315789	0.215827	0.102041
3	0.666667	0.416667	0.333333	0.789474	0.575540	0.816327
4	0.111111	0.000000	0.833333	0.157895	0.000000	0.163265
...	...	...	...	...	...	...
95	0.777778	0.750000	1.000000	0.736842	0.273381	0.448980
96	0.222222	0.666667	0.500000	0.052632	0.712230	0.489796
97	0.000000	0.854167	0.166667	0.473684	0.230216	0.693878
98	0.777778	0.895833	0.000000	0.052632	0.719424	0.816327
99	0.222222	0.479167	0.500000	0.842105	0.158273	0.591837

100 rows × 6 columns

```
data_new.head()
```

	Fan Challenges Completed	Predictive Accuracy (%)	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)	Time on Live 360 (mins)	Real-Time Chat Activity (Messages Sent)
0	0.444444	0.625000	0.500000	0.526316	0.431655	0.408163
1	0.777778	0.208333	0.166667	0.421053	0.287770	0.714286
2	0.222222	0.833333	0.000000	0.315789	0.215827	0.102041
3	0.666667	0.416667	0.333333	0.789474	0.575540	0.816327
4	0.111111	0.000000	0.833333	0.157895	0.000000	0.163265

```
def find_outliers_iqr(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    # Define outliers as values below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR
    outliers = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
    return outliers
outliers = find_outliers_iqr(data[columns_to_check])


print(data[outliers.any(axis=1)])
```

```
Empty DataFrame
Columns: [User ID, Fan Challenges Completed, Predictive Accuracy (%), Virtual Merchandise Purchases, Sponsorship Interaction
Index: []
```

correlation matrix to identify relationships

```
correlations = data_new.corr()
```

```
correlations[['Virtual Merchandise Purchases',
              'Sponsorship Interactions (Ad Clicks)']].sort_values(by='Virtual Merchandise Purchases', ascending=False)
```

	Virtual Merchandise Purchases	Sponsorship Interactions (Ad Clicks)
Virtual Merchandise Purchases	1.000000	0.070550
Fan Challenges Completed	0.159378	-0.065239
Sponsorship Interactions (Ad Clicks)	0.070550	1.000000
Predictive Accuracy (%)	0.022194	0.056612
Time on Live 360 (mins)	-0.007527	-0.073929
Real-Time Chat Activity (Messages Sent)	-0.044676	0.191292

Preparing the data for Modeling

```
from sklearn.model_selection import train_test_split
```

```
data_new['Purchased'] = data_new['Virtual Merchandise Purchases'] > 0
```

```
X = data_new[['Fan Challenges Completed',
                  'Predictive Accuracy (%)',
                  'Sponsorship Interactions (Ad Clicks)',
                  'Time on Live 360 (mins)',
                  'Real-Time Chat Activity (Messages Sent)']]
y = data_new['Purchased']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training a LogisticRegression Model

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import classification_report
```

```
model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

```
y_pred = model.predict(X_test)
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.00      0.00      0.00         2
        True       0.90      1.00      0.95        18

    accuracy                           0.90        20
   macro avg       0.45      0.50      0.47        20
weighted avg       0.81      0.90      0.85        20

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[ 0  2]
 [ 0 18]]
```

Correlation Analysis

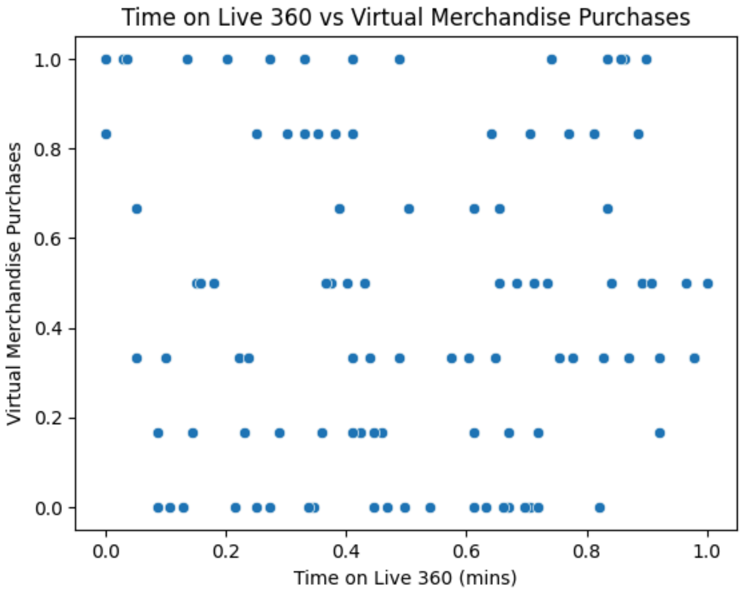
```
activities = ['Fan Challenges Completed', 'Time on Live 360 (mins)',
              'Real-Time Chat Activity (Messages Sent)']


targets = ['Virtual Merchandise Purchases', 'Sponsorship Interactions (Ad Clicks)']



corr_activities_targets = data_new[activities + targets].corr()



print("Correlation between user activities and merchandise purchases/sponsorship interactions:")
print(corr_activities_targets[['Virtual Merchandise Purchases', 'Sponsorship Interactions (Ad Clicks)']].loc[activities])
```

```
Correlation between user activities and merchandise purchases/sponsorship interactions:
                                         Virtual Merchandise Purchases  \
Fan Challenges Completed                                      0.159378
Time on Live 360 (mins)                                      -0.007527
Real-Time Chat Activity (Messages Sent)                      -0.044676

                                         Sponsorship Interactions (Ad Clicks)
Fan Challenges Completed                                            -0.065239
Time on Live 360 (mins)                                             -0.073929
Real-Time Chat Activity (Messages Sent)                              0.191292
```

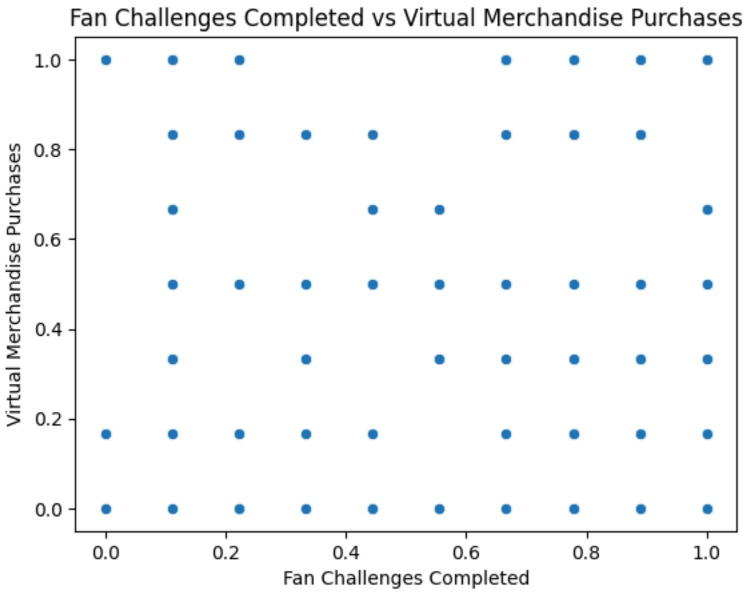
```
import seaborn as sns


import matplotlib.pyplot as plt
```

User Activities and Purchases

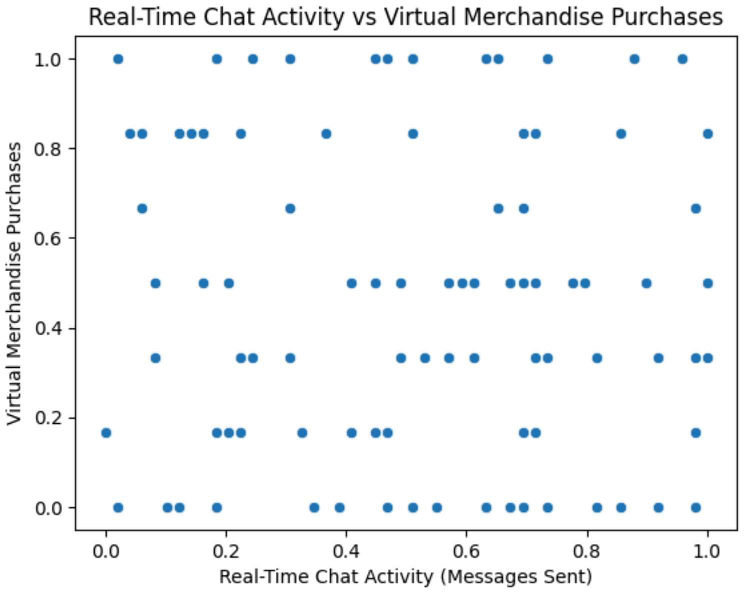
```
sns.scatterplot(x='Time on Live 360 (mins)', y='Virtual Merchandise Purchases', data=data_new)
plt.title('Time on Live 360 vs Virtual Merchandise Purchases')
plt.show()
```



```
sns.scatterplot(x='Fan Challenges Completed', y='Virtual Merchandise Purchases', data=data_new)
plt.title('Fan Challenges Completed vs Virtual Merchandise Purchases')
plt.show()
```

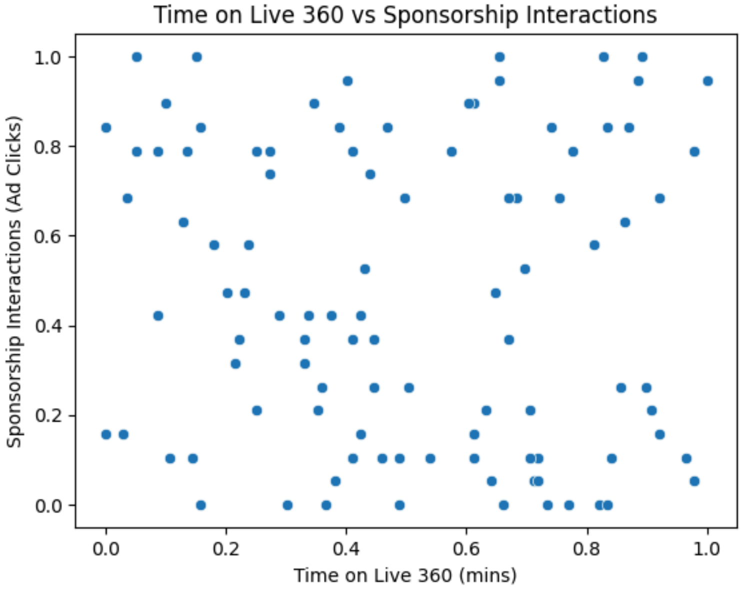


```
sns.scatterplot(x='Real-Time Chat Activity (Messages Sent)', y='Virtual Merchandise Purchases', data=data_new)
plt.title('Real-Time Chat Activity vs Virtual Merchandise Purchases')
plt.show()
```

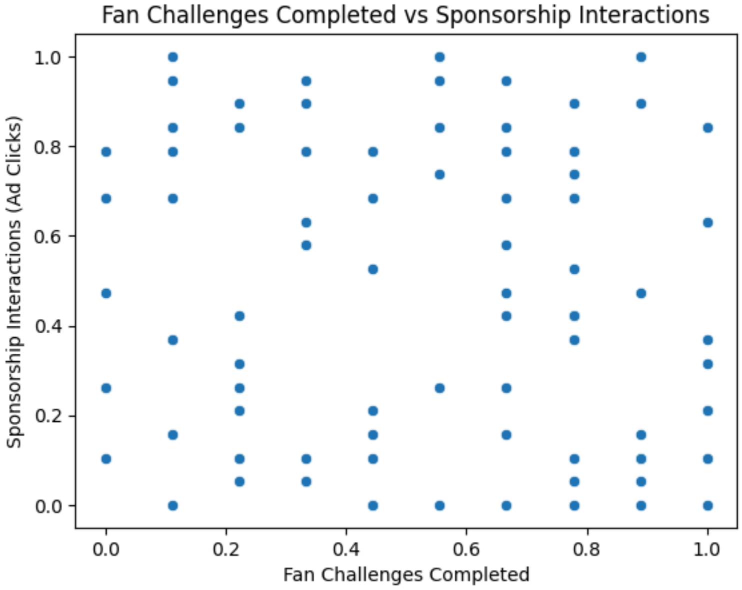


User Activities and Sponsorship Interactions

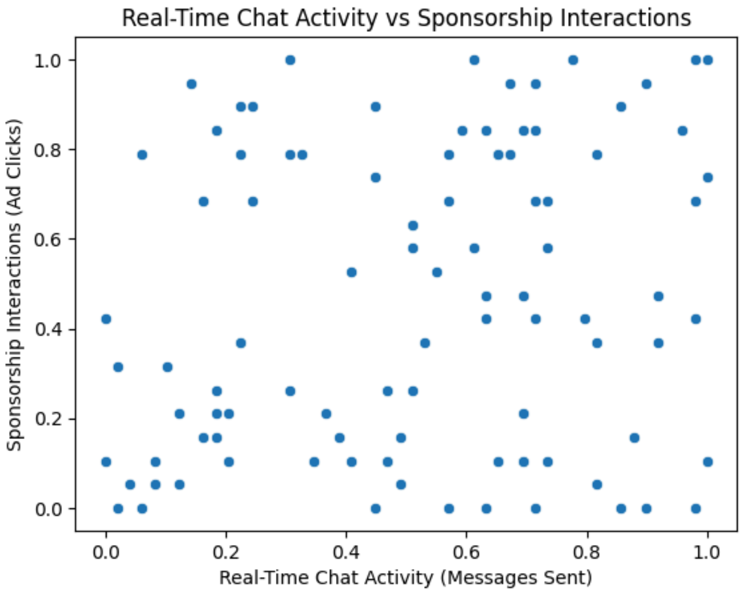
```
sns.scatterplot(x='Time on Live 360 (mins)', y='Sponsorship Interactions (Ad Clicks)', data=data_new)
plt.title('Time on Live 360 vs Sponsorship Interactions')
plt.show()
```



```
sns.scatterplot(x='Fan Challenges Completed', y='Sponsorship Interactions (Ad Clicks)', data=data_new)
plt.title('Fan Challenges Completed vs Sponsorship Interactions')
plt.show()
```

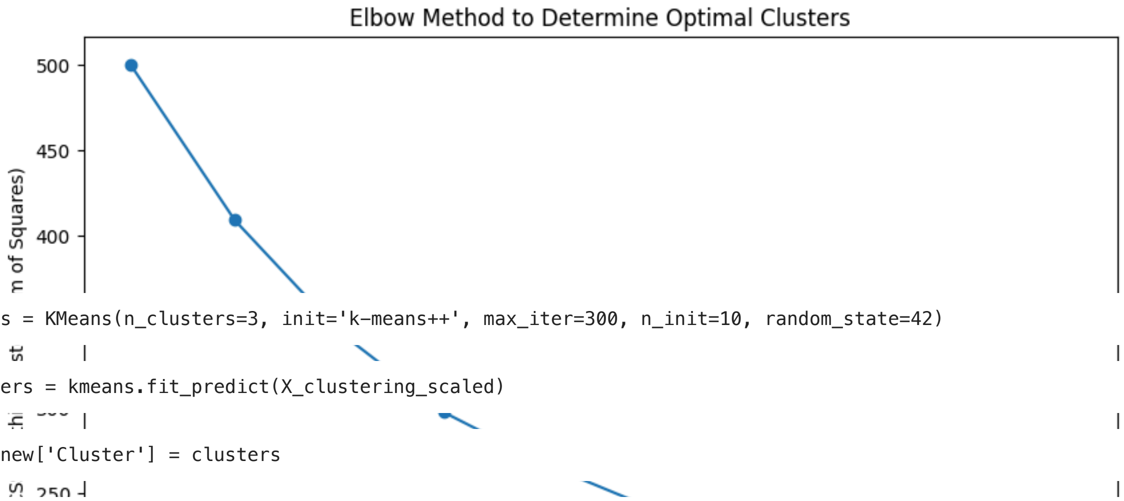


```
sns.scatterplot(x='Real-Time Chat Activity (Messages Sent)', y='Sponsorship Interactions (Ad Clicks)', data=data_new)
plt.title('Real-Time Chat Activity vs Sponsorship Interactions')
plt.show()
```

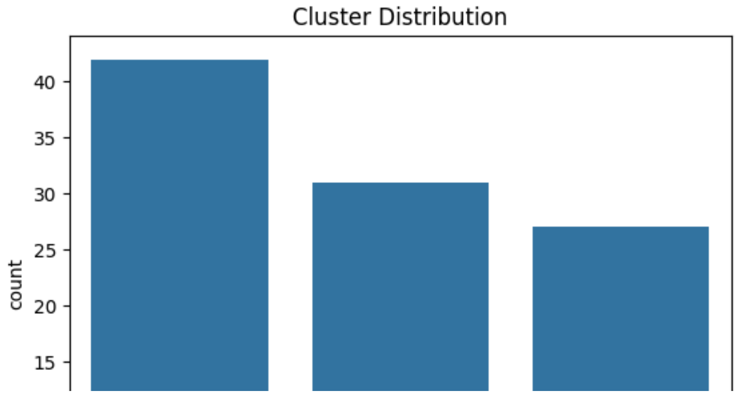


Apply K-Means Clustering

```
from sklearn.cluster import KMeans


X_clustering = data_new[['Fan Challenges Completed', 'Time on Live 360 (mins)',
                             'Real-Time Chat Activity (Messages Sent)', 'Virtual Merchandise Purchases',
                             'Sponsorship Interactions (Ad Clicks)']]


from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()


X_clustering_scaled = scaler.fit_transform(X_clustering)


wcss = []


for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(X_clustering_scaled)
    wcss.append(kmeans.inertia_)


plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method to Determine Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within Cluster Sum of Squares)')
plt.show()
```



```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=42)

clusters = kmeans.fit_predict(X_clustering_scaled)

data_new['Cluster'] = clusters

sns.countplot(x='Cluster', data=data_new)
plt.title('Cluster Distribution')
plt.show()
```


Cluster Distribution
count