Abstract

A key challenge for e-commerce businesses is to analyze the trend in the market to increase their sales. The trend can be easily observed if the companies can group the customers; based on their activity on the e-commerce site. This grouping can be done by applying different criteria like previous orders, mostly searched brands and so on.

Problem Statement

Given the e-commerce data, use k-means clustering algorithm to cluster customers with similar interest.

Dataset Information

The data was collected from a well known e-commerce website over a period of time based on the customer's search profile.

Scope

● Analyzing the existing customer data and getting valuable insights about the purchase pattern ● Data pre-processing including missing value treatment ● Segmenting customer based on the optimum number of clusters ('k') with the help of silhouette score

Data Definition

Column Description

1. Cust_ID Unique numbering for customers
2. Gender Gender of the customer
3. Orders Number of orders placed by each customer in the past

Remaining 35 features (brands) contains the number of times customers have searched them

Content

1. Import Packages

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

from sklearn.preprocessing import MinMaxScaler
```

2. Read Data

In [2]:
```
data = pd.read_excel(r'C:\Users\navin\OneDrive\Desktop\WorkTree\sample_table\cust_data.xlsx')
```

In [3]:
```
data.head()
```

Out[3]:

	Cust_ID	Gender	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	...	LG	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	Microso
0	1	M	7	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	2	F	0	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	
2	3	M	7	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	4	F	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	5	NaN	10	0	0	0	0	0	0	0	...	0	0	2	0	0	0	0	

5 rows × 38 columns

3. Understand and Prepare the Data

3.1 Data Types and Dimensions

In [4]:
```
data.dtypes
```

Out[4]:
```
Cust_ID                   int64
Gender                   object
Orders                    int64
Jordan                    int64
Gatorade                  int64
Samsung                   int64
Asus                      int64
Udis                      int64
Mondelez International    int64
Wrangler                  int64
Vans                      int64
Fila                      int64
Brooks                    int64
H&M                       int64
Dairy Queen               int64
Fendi                     int64
Hewlett Packard           int64
Pladis                    int64
Asics                     int64
Siemens                   int64
J.M. Smucker              int64
Pop Chips                 int64
Juniper                   int64
Huawei                    int64
Compaq                    int64
IBM                       int64
Burberry                  int64
Mi                        int64
LG                        int64
Dior                      int64
Scabal                    int64
Tommy Hilfiger            int64
Hollister                 int64
Forever 21                int64
Colavita                  int64
Microsoft                 int64
Jiffy mix                 int64
Kraft                     int64
dtype: object
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 38 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Cust_ID                  30000 non-null  int64
 1   Gender                   27276 non-null  object
 2   Orders                   30000 non-null  int64
 3   Jordan                   30000 non-null  int64
 4   Gatorade                 30000 non-null  int64
 5   Samsung                  30000 non-null  int64
 6   Asus                     30000 non-null  int64
 7   Udis                     30000 non-null  int64
 8   Mondelez International   30000 non-null  int64
 9   Wrangler                 30000 non-null  int64
 10  Vans                     30000 non-null  int64
 11  Fila                     30000 non-null  int64
 12  Brooks                   30000 non-null  int64
 13  H&M                      30000 non-null  int64
 14  Dairy Queen              30000 non-null  int64
 15  Fendi                    30000 non-null  int64
 16  Hewlett Packard          30000 non-null  int64
 17  Pladis                   30000 non-null  int64
 18  Asics                    30000 non-null  int64
 19  Siemens                  30000 non-null  int64
 20  J.M. Smucker             30000 non-null  int64
 21  Pop Chips                30000 non-null  int64
 22  Juniper                  30000 non-null  int64
 23  Huawei                   30000 non-null  int64
 24  Compaq                   30000 non-null  int64
 25  IBM                      30000 non-null  int64
 26  Burberry                 30000 non-null  int64
 27  Mi                       30000 non-null  int64
 28  LG                       30000 non-null  int64
 29  Dior                     30000 non-null  int64
 30  Scabal                   30000 non-null  int64
 31  Tommy Hilfiger           30000 non-null  int64
 32  Hollister                30000 non-null  int64
 33  Forever 21               30000 non-null  int64
 34  Colavita                 30000 non-null  int64
 35  Microsoft                30000 non-null  int64
 36  Jiffy mix                30000 non-null  int64
 37  Kraft                    30000 non-null  int64
dtypes: int64(37), object(1)
memory usage: 8.7+ MB
```

3.2 Distribution of Variables

In [6]:
```
data.shape
```

Out[6]:
```
(30000, 38)
```

3.3 Statistical Summary

In [7]:
```
data.describe()
```

Out[7]:

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	...	30000.0000
mean	15000.500000	4.169800	0.267433	0.252333	0.222933	0.161333	0.143533	0.139767	0.106933	0.111433	...	0.1025
std	8660.398374	3.590311	0.804778	0.705368	0.917494	0.740038	0.641258	0.525840	0.515921	0.547990	...	0.4863
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
25%	7500.750000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
50%	15000.500000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
75%	22500.250000	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
max	30000.000000	12.000000	24.000000	15.000000	27.000000	17.000000	14.000000	31.000000	9.000000	16.000000	...	19.0000

8 rows × 37 columns

In [8]:
```
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Cust_ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
Orders	30000.0	4.169800	3.590311	0.0	1.00	4.0	7.00	12.0
Jordan	30000.0	0.267433	0.804778	0.0	0.00	0.0	0.00	24.0
Gatorade	30000.0	0.252333	0.705368	0.0	0.00	0.0	0.00	15.0
Samsung	30000.0	0.222933	0.917494	0.0	0.00	0.0	0.00	27.0
Asus	30000.0	0.161333	0.740038	0.0	0.00	0.0	0.00	17.0
Udis	30000.0	0.143533	0.641258	0.0	0.00	0.0	0.00	14.0
Mondelez International	30000.0	0.139767	0.525840	0.0	0.00	0.0	0.00	31.0
Wrangler	30000.0	0.106933	0.515921	0.0	0.00	0.0	0.00	9.0
Vans	30000.0	0.111433	0.547990	0.0	0.00	0.0	0.00	16.0
Fila	30000.0	0.094267	0.531592	0.0	0.00	0.0	0.00	15.0
Brooks	30000.0	0.140133	0.471278	0.0	0.00	0.0	0.00	12.0
H&M	30000.0	0.328200	0.807655	0.0	0.00	0.0	0.00	18.0
Dairy Queen	30000.0	0.209333	1.116820	0.0	0.00	0.0	0.00	114.0
Fendi	30000.0	0.141467	0.529210	0.0	0.00	0.0	0.00	18.0
Hewlett Packard	30000.0	0.161100	0.604835	0.0	0.00	0.0	0.00	10.0
Pladis	30000.0	0.106100	0.516512	0.0	0.00	0.0	0.00	26.0
Asics	30000.0	0.300233	1.119167	0.0	0.00	0.0	0.00	66.0
Siemens	30000.0	0.048067	0.338763	0.0	0.00	0.0	0.00	11.0
J.M. Smucker	30000.0	0.754800	1.262166	0.0	0.00	0.0	1.00	64.0
Pop Chips	30000.0	0.247900	0.724106	0.0	0.00	0.0	0.00	21.0
Juniper	30000.0	0.470833	1.346159	0.0	0.00	0.0	1.00	79.0
Huawei	30000.0	0.258000	0.848288	0.0	0.00	0.0	0.00	44.0
Compaq	30000.0	0.118767	0.597612	0.0	0.00	0.0	0.00	30.0
IBM	30000.0	0.031967	0.264475	0.0	0.00	0.0	0.00	11.0
Burberry	30000.0	0.428033	1.098876	0.0	0.00	0.0	0.00	37.0
Mi	30000.0	0.121333	0.478977	0.0	0.00	0.0	0.00	9.0
LG	30000.0	0.102533	0.486376	0.0	0.00	0.0	0.00	19.0
Dior	30000.0	0.271133	0.714682	0.0	0.00	0.0	0.00	12.0
Scabal	30000.0	0.370067	0.758465	0.0	0.00	0.0	1.00	11.0
Tommy Hilfiger	30000.0	0.158967	0.510527	0.0	0.00	0.0	0.00	8.0
Hollister	30000.0	0.077667	0.383370	0.0	0.00	0.0	0.00	9.0
Forever 21	30000.0	0.057333	0.300082	0.0	0.00	0.0	0.00	8.0
Colavita	30000.0	0.192200	0.641306	0.0	0.00	0.0	0.00	22.0
Microsoft	30000.0	0.116367	0.446578	0.0	0.00	0.0	0.00	14.0
Jiffy mix	30000.0	0.088033	0.399277	0.0	0.00	0.0	0.00	8.0
Kraft	30000.0	0.070900	0.387915	0.0	0.00	0.0	0.00	16.0

3.4 Duplicated Value

In [9]:
```
data.duplicated().sum()
```

Out[9]:
```
0
```

3.5 Missing Data Treatment

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
Cust_ID                    0
Gender                  2724
Orders                     0
Jordan                     0
Gatorade                   0
Samsung                    0
Asus                       0
Udis                       0
Mondelez International     0
Wrangler                   0
Vans                       0
Fila                       0
Brooks                     0
H&M                        0
Dairy Queen                0
Fendi                      0
Hewlett Packard            0
Pladis                     0
Asics                      0
Siemens                    0
J.M. Smucker               0
Pop Chips                  0
Juniper                    0
Huawei                     0
Compaq                     0
IBM                        0
Burberry                   0
Mi                         0
LG                         0
Dior                       0
Scabal                     0
Tommy Hilfiger             0
Hollister                  0
Forever 21                 0
Colavita                   0
Microsoft                  0
Jiffy mix                  0
Kraft                      0
dtype: int64
```

In [11]:
```
data['Gender'].unique()
```

Out[11]:
```
array(['M', 'F', nan], dtype=object)
```

In [12]:
```
data['Gender'].value_counts(dropna = False)
```

Out[12]:
```
F      22054
M       5222
NaN     2724
Name: Gender, dtype: int64
```

In [13]:
```
data['Gender'].mode()
```

Out[13]:
```
0    F
Name: Gender, dtype: object
```

In [14]:
```
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
```

In [15]:
```
data['Gender'].value_counts(dropna = False)
```

Out[15]:
```
F    24778
M     5222
Name: Gender, dtype: int64
```

In [16]:
```
data.isnull().sum()
```

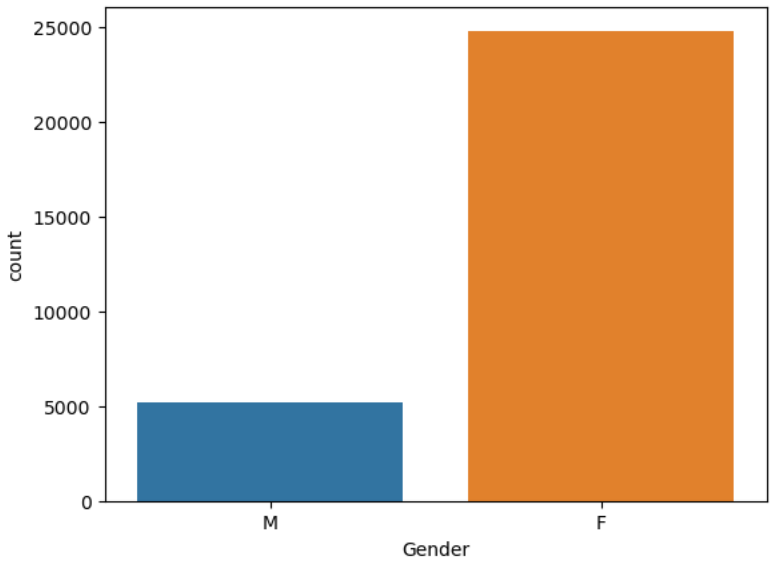
Out[16]:
```
Cust_ID                  0
Gender                   0
Orders                   0
Jordan                   0
Gatorade                 0
Samsung                  0
Asus                     0
Udis                     0
Mondelez International    0
Wrangler                 0
Vans                     0
Fila                     0
Brooks                   0
H&M                      0
Dairy Queen              0
Fendi                    0
Hewlett Packard          0
Pladis                   0
Asics                    0
Siemens                  0
J.M. Smucker             0
Pop Chips                0
Juniper                  0
Huawei                   0
Compaq                   0
IBM                      0
Burberry                 0
Mi                       0
LG                       0
Dior                     0
Scabal                   0
Tommy Hilfiger           0
Hollister                0
Forever 21               0
Colavita                 0
Microsoft                0
Jiffy mix                0
Kraft                    0
dtype: int64
```

3.6 Visualization

3.6.1 Distribution of data among Genders

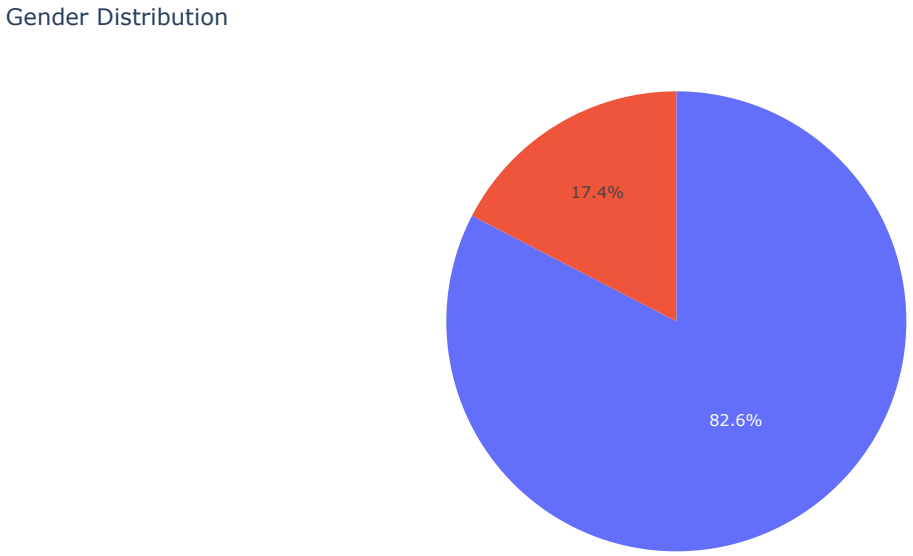
In [17]:
```
sns.countplot(data, x ='Gender')
```

Out[17]:
```
<Axes: xlabel='Gender', ylabel='count'>
```



3.6.2 Distubution of data among Genders in a pie chart

In [18]:
```
fig1 = px.pie(data, names = 'Gender',  title= 'Gender Distribution')
fig1.show()
```



In [19]:
```
data.head()
```

Out[19]:

	Cust_ID	Gender	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	...	LG	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	Microso
0	1	M	7	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	2	F	0	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	
2	3	M	7	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	4	F	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	5	F	10	0	0	0	0	0	0	0	...	0	0	2	0	0	0	0	

5 rows × 38 columns

3.6.3 Total search made by each customer in the e-commerce platform

In [20]:
```
total_search = data.iloc[:,3:].sum(axis = 1)
result_df = data[['Cust_ID','Gender','Orders']].copy()
result_df['total_search'] = total_search
result_df
```

Out[20]:

	Cust_ID	Gender	Orders	total_search
0	1	M	7	2
1	2	F	0	18
2	3	M	7	5
3	4	F	0	2
4	5	F	10	16
...	...	...	...	...
29995	29996	M	0	1
29996	29997	M	1	1
29997	29998	M	0	2
29998	29999	M	0	1
29999	30000	F	3	5

30000 rows × 4 columns

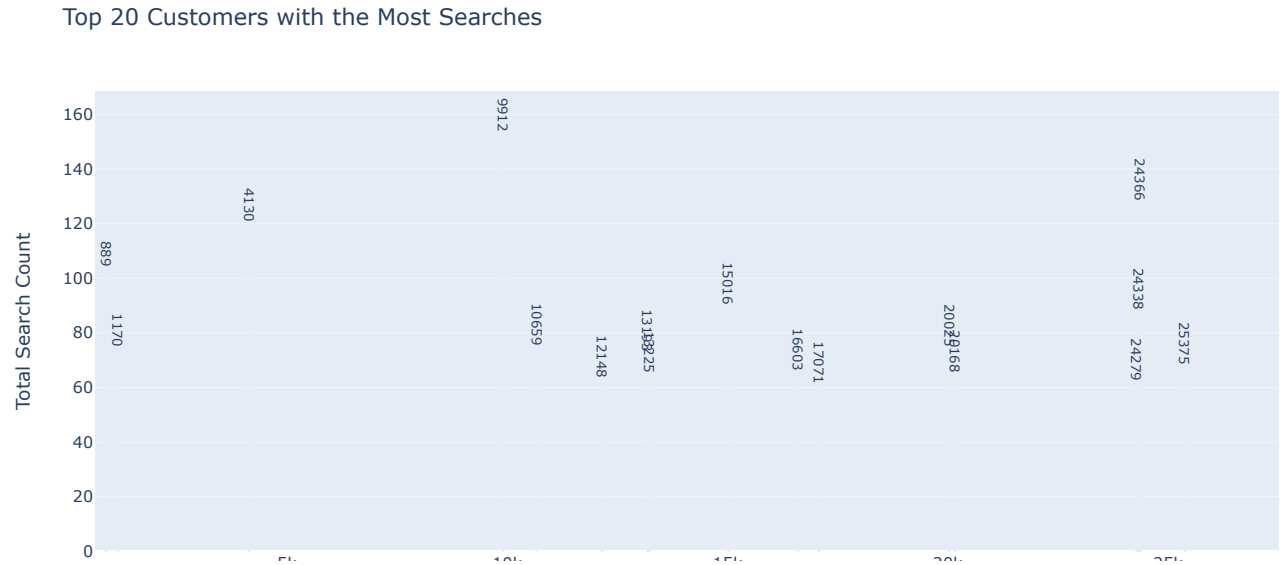
3.6.4 Top 20 customers with maximum of searches

In [21]:
```
top_20_customers = result_df.nlargest(20, 'total_search')
top_20_customers
```

Out[21]:

	Cust_ID	Gender	Orders	total_search
9911	9912	F	2	160
24365	24366	F	2	136
4129	4130	F	1	127
888	889	F	0	109
15015	15016	F	10	98
24337	24338	M	11	96
29583	29584	F	11	87
29131	29132	F	1	85
10658	10659	F	0	83
20024	20025	F	0	83
1169	1170	F	0	81
13192	13193	F	1	81
28887	28888	F	4	79
25374	25375	F	2	76
16602	16603	F	10	74
13224	13225	F	6	73
20167	20168	F	2	73
12147	12148	F	10	71
24278	24279	F	11	70
17070	17071	F	11	69

In [22]:
```
fig = px.bar(
    top_20_customers,
    x='Cust_ID',
    y='total_search',
    title='Top 20 Customers with the Most Searches',
    labels={'Cust_ID': 'Customer ID', 'total_search': 'Total Search Count'},
    color_discrete_sequence=['red'],
)
for i, row in top_20_customers.iterrows():
    fig.add_annotation(
        x=row['Cust_ID'],
        y=row['total_search'],
        text=row['Cust_ID'],
        showarrow=False,
        font=dict(size=10),
        textangle=90,
        xanchor='center',
    )
fig.show()
```



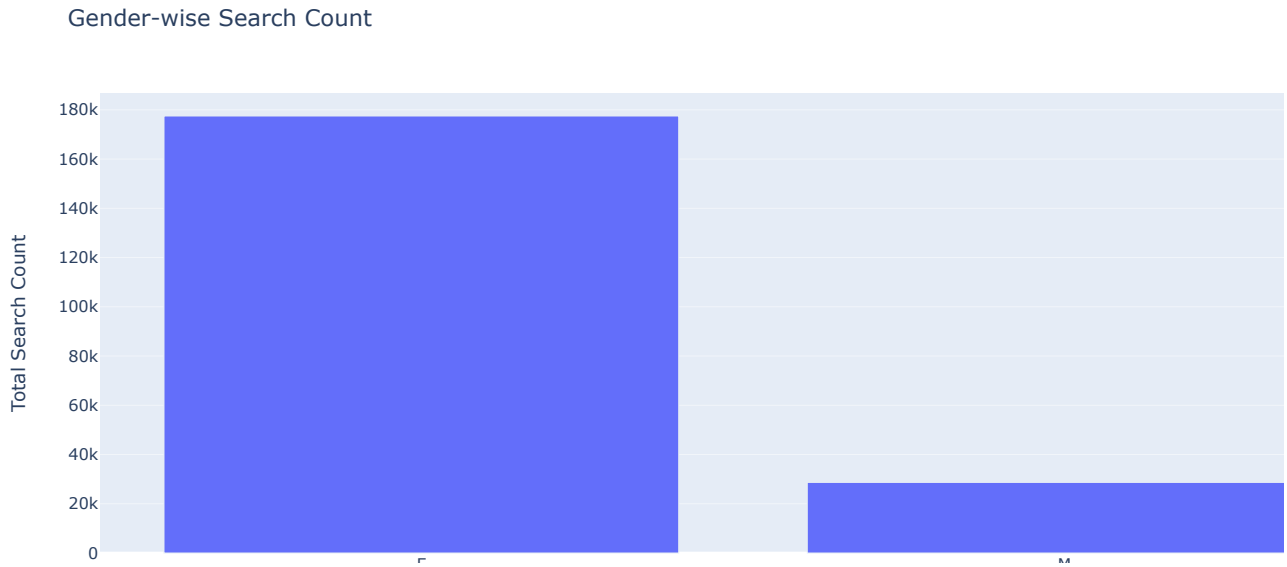
3.6.5 Gender wise total_search count

In [23]:
```
gender_wise_search_count_data = result_df.groupby('Gender')['total_search'].sum().reset_index()
gender_wise_search_count_data
```

Out[23]:

	Gender	total_search
0	F	177491
1	M	28652

In [24]:
```
fig = px.bar(
    gender_wise_search_count_data,
    x='Gender',
    y='total_search',
    title='Gender-wise Search Count',
    labels={'Gender': 'Gender', 'total_search': 'Total Search Count'},
)
fig.show()
```



3.6.6 Brand wise total search count

In [25]:
```
data.iloc[:,3:]
```

Out[25]:

	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	Fila	Brooks	...	LG	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	Microsoft
0	0	0	0	0	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	2	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0
29996	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
29997	0	1	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0
29998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
29999	2	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

30000 rows × 35 columns

In [26]:
```
brand_df = data.iloc[:,3:].sum().sort_values(ascending = False).reset_index()
brand_df.columns = ['Brand', 'Total_Search_Count']
brand_df
```

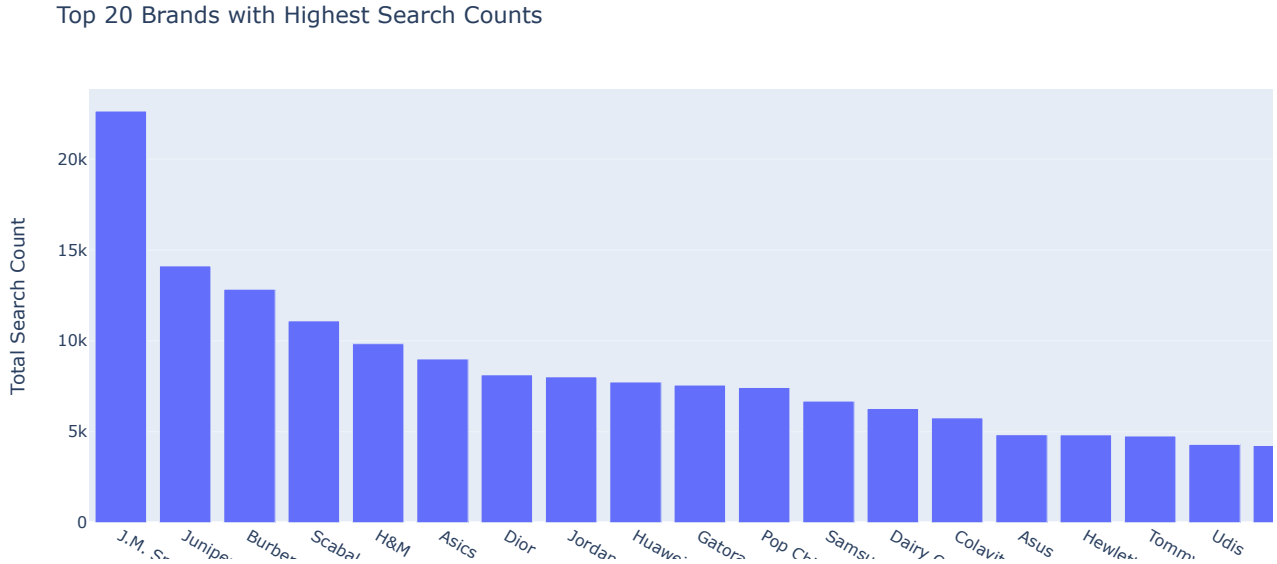
Out[26]:

	Brand	Total_Search_Count
0	J.M. Smucker	22644
1	Juniper	14125
2	Burberry	12841
3	Scabal	11102
4	H&M	9846
5	Asics	9007
6	Dior	8134
7	Jordan	8023
8	Huawei	7740
9	Gatorade	7570
10	Pop Chips	7437
11	Samsung	6688
12	Dairy Queen	6280
13	Colavita	5766
14	Asus	4840
15	Hewlett Packard	4833
16	Tommy Hilfiger	4769
17	Udis	4306
18	Fendi	4244
19	Brooks	4204
20	Mondelez International	4193
21	Mi	3640
22	Compaq	3563
23	Microsoft	3491
24	Vans	3343
25	Wrangler	3208
26	Pladis	3183
27	LG	3076
28	Fila	2828
29	Jiffy mix	2641
30	Hollister	2330
31	Kraft	2127
32	Forever 21	1720
33	Siemens	1442
34	IBM	959

3.6.7 Top 20 brand names with highest total search count

In [27]:
```
top_20_brands = brand_df.head(20)
```

In [28]:
```
fig = px.bar(
    top_20_brands,
    x='Brand',
    y='Total_Search_Count',
    title='Top 20 Brands with Highest Search Counts',
    labels={'Brand': 'Brand', 'Total_Search_Count': 'Total Search Count'},
)
fig.show()
```



4. Scaling

Using min max scaler

In [29]:
```
data_to_scale = data.iloc[:, 3: ]
minmax_scaler = MinMaxScaler()
minmax_scaled_data = minmax_scaler.fit_transform(data_to_scale)
minmax_scaled_data
```

Out[29]:
```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.06666667, 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.06666667, 0.        , ..., 0.07142857, 0.        ,
        0.        ],
       ...,
       [0.        , 0.06666667, 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.08333333, 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ]])
```

In [30]:
```
# Create Dataframe for scaled data
minmax_scaled_data_df = pd.DataFrame(minmax_scaled_data,columns = data_to_scale.columns)
minmax_scaled_data_df
```

Out[30]:

	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	Fila	Brooks	...	LG	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Cola
0	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.1250	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
1	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.083333	0.000000	0.0	0.0	0.0	
2	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
3	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
4	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.181818	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.083333	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
29996	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
29997	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0625	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
29998	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	
29999	0.083333	0.000000	0.0	0.058824	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.0	0.000000	0.000000	0.0	0.0	0.0	

30000 rows × 35 columns

5. Silhoutte Score

In [31]:
```
best_score = -1
best_k = -1
best_labels = None

silhouette_scores = []

for n_clusters in range(2,9):
    # Perform clustering using K-means
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=10)
    labels = kmeans.fit_predict(minmax_scaled_data)

    # Calculate the Silhouette Score
    silhouette_avg = silhouette_score(minmax_scaled_data, labels)
    silhouette_scores.append(silhouette_avg)

    # Print the Silhouette Score for each number of clusters
    print(f"Number of Clusters: {n_clusters} - Silhouette Score: {silhouette_avg}")

    # Check if the current Silhouette Score is the best
    if silhouette_avg > best_score:
        best_score = silhouette_avg
        best_k = n_clusters
        best_labels = labels

# Print the best k value
print(f"\nBest k value: {best_k} (Silhouette Score: {best_score})")
```

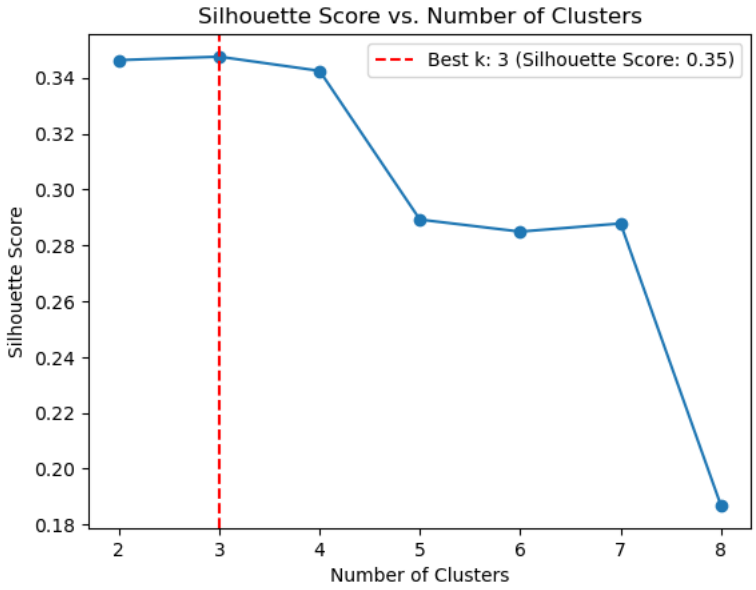
```
Number of Clusters: 2 - Silhouette Score: 0.3463431705485548
Number of Clusters: 3 - Silhouette Score: 0.347580478203035
Number of Clusters: 4 - Silhouette Score: 0.34250345488796646
Number of Clusters: 5 - Silhouette Score: 0.28923913940598434
Number of Clusters: 6 - Silhouette Score: 0.28492093064218355
Number of Clusters: 7 - Silhouette Score: 0.2878383516092045
Number of Clusters: 8 - Silhouette Score: 0.18686583010174967

Best k value: 3 (Silhouette Score: 0.347580478203035)
```

In [32]:
```
best_labels
```

Out[32]:
```
array([0, 1, 0, ..., 0, 0, 0])
```

5.1 Silhouette Score Analysis for Optimal Number of Clusters

In [33]:
```
# Plot the line graph
plt.plot(range(2, 9), silhouette_scores, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs. Number of Clusters')

# Highlight the best k value
plt.axvline(x=best_k, color='r', linestyle='--', label=f'Best k: {best_k} (Silhouette Score: {best_score:.2f})')
plt.legend()

# Display the plot
plt.show()
```

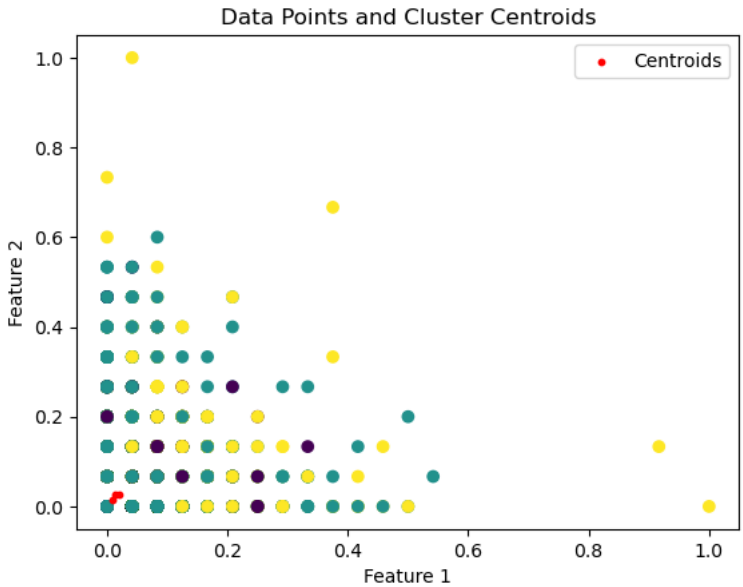


6. K-Means Clustering with 3 Clusters: Cluster Centers and Assignments

In [34]:
```
# Fit the K-Means model to your data
kmeans = KMeans(n_clusters=3, random_state =0, n_init = 10)
kmeans.fit(minmax_scaled_data)

 # Get the cluster centroids and Labels
cluster_centers = kmeans.cluster_centers_
cluster_assignments = kmeans.labels_
```

Data Points and Cluster Centroids Visualization

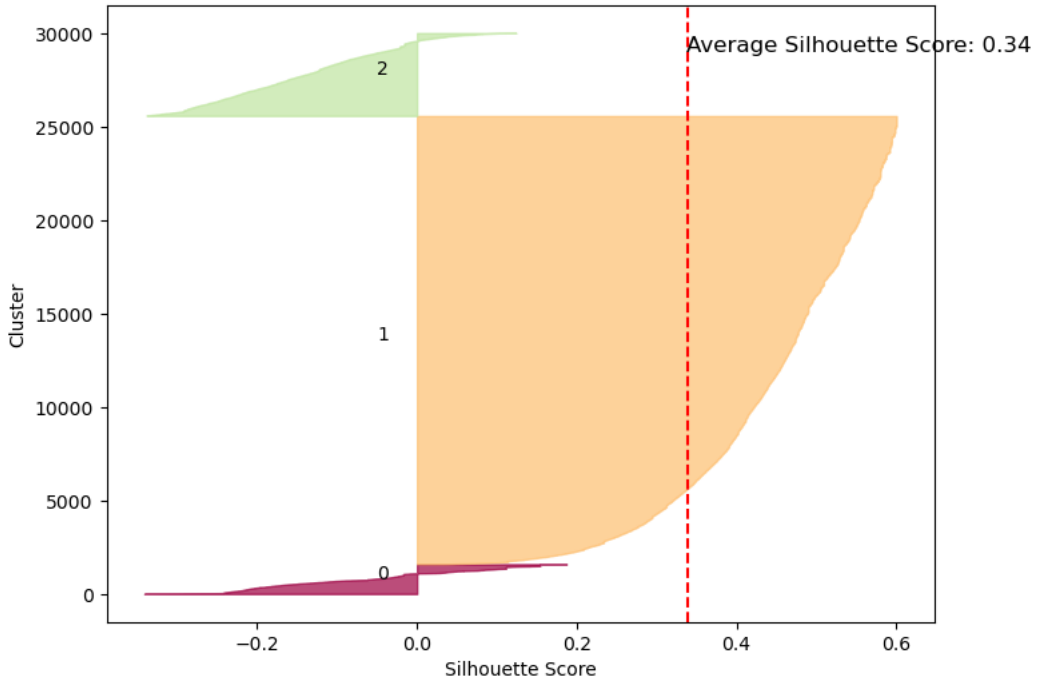
In [35]:
```
plt.scatter(minmax_scaled_data[:, 0], minmax_scaled_data[:, 1], c=cluster_assignments, cmap='viridis')
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], s=10, c='red', label='Centroids')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('Data Points and Cluster Centroids')
plt.show()
```



7. Silhoutte Analysis for Cluster Quality

Silhouette analysis is a method to assess the quality of clusters created by a clustering algorithm. It measures how distinct and well-separated the clusters are. A higher silhouette score indicates better-defined clusters(well-separated), while a lower score(overlap or don't look clear) suggests that the clusters may not be meaningful or well-separated.

In [36]:
```
from sklearn.metrics import silhouette_samples
from sklearn.metrics import silhouette_score

silhouette_samples_scores = silhouette_samples(minmax_scaled_data, cluster_assignments)
average_silhouette_score = silhouette_score(minmax_scaled_data, cluster_assignments)

# Create a horizontal bar plot for each data point
y_lower = 10  # Initialize the lower y-coordinate

# Create a subplot
fig, ax1 = plt.subplots(1,1)
fig.set_size_inches(8, 6)


# Get a colormap using matplotlib.colormaps.get_cmap
cmap = plt.get_cmap("Spectral")

# Loop through each cluster to plot the silhouette plot
for i in range(len(np.unique(cluster_assignments))):
    cluster_data = silhouette_samples_scores[cluster_assignments == i]
    cluster_data.sort()

    size_cluster_i = cluster_data.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cmap(float(i) / len(np.unique(cluster_assignments)))

    ax1.fill_betweenx(
        np.arange(y_lower, y_upper),
        0,
        cluster_data,
        facecolor=color,
        edgecolor=color,
        alpha=0.7,
    )

    # Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for the next plot
    y_lower = y_upper + 10  # 10 for the 0 samples

ax1.set_xlabel("Silhouette Score")
ax1.set_ylabel("Cluster")

# The vertical line for the average silhouette score of all the values
ax1.axvline(x=average_silhouette_score, color="red", linestyle="--")
ax1.text(0.7, 0.95, f'Average Silhouette Score: {average_silhouette_score:.2f}', transform=ax1.transAxes, fontsize=12, verticalalignment=

plt.show()
```



8. Cluster Analysis: Exploring Individual Clusters

In [37]:
```
cluster_assignments
```

Out[37]:
```
array([1, 2, 1, ..., 1, 1, 1])
```

Adding Cluster Labels to scaled Data

In [38]:
```
cluster_analysis_data = minmax_scaled_data_df.copy()
cluster_analysis_data['Cluster labels'] = cluster_assignments
cluster_analysis_data
```

Out[38]:

	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	Fila	Brooks	...	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita
0	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.1250	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
1	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.083333	0.000000	0.0	0.0	0.0	0.0
2	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
3	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
4	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.181818	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.083333	...	0.000000	0.000000	0.0	0.0	0.0	0.0
29996	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
29997	0.000000	0.066667	0.0	0.000000	0.0	0.0	0.0	0.0625	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
29998	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0
29999	0.083333	0.000000	0.0	0.058824	0.0	0.0	0.0	0.0000	0.0	0.000000	...	0.000000	0.000000	0.0	0.0	0.0	0.0

30000 rows × 36 columns

Adding Cluster Labels to Original Data

In [39]:
```
final_cluster_data = data.copy()
final_cluster_data['Cluster labels'] = cluster_assignments
final_cluster_data
```

Out[39]:

	Cust_ID	Gender	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	...	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	Microsc
0	1	M	7	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	2	F	0	0	1	0	0	0	0	0	...	1	0	0	0	0	0	
2	3	M	7	0	1	0	0	0	0	0	...	0	0	0	0	0	0	
3	4	F	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	5	F	10	0	0	0	0	0	0	0	...	0	2	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	M	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
29996	29997	M	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	
29997	29998	M	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	
29998	29999	M	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
29999	30000	F	3	2	0	0	1	0	0	0	...	0	0	0	0	0	0	

30000 rows × 39 columns

Analysis of each Clusters

In [40]:
```
final_cluster_data['Cluster labels'].unique()
```

Out[40]:
```
array([1, 2, 0])
```

In [41]:
```
final_cluster_data['Cluster labels'].value_counts()
```

Out[41]:
```
1    23979
2     4430
0     1591
Name: Cluster labels, dtype: int64
```

8.1 Cluster 1 Analysis

In [42]:
```
# Filter the data to select all rows belonging to Cluster 1
cluster_1 = final_cluster_data[final_cluster_data['Cluster labels'] == 0]
```

In [43]:
```
# Calculate and display the count of brands in Cluster 1
cluster_1_df = cluster_1.iloc[:,3:-1].sum().sort_values(ascending = False).reset_index()
cluster_1_df.columns = ['Brand', 'count']
cluster_1_df
```

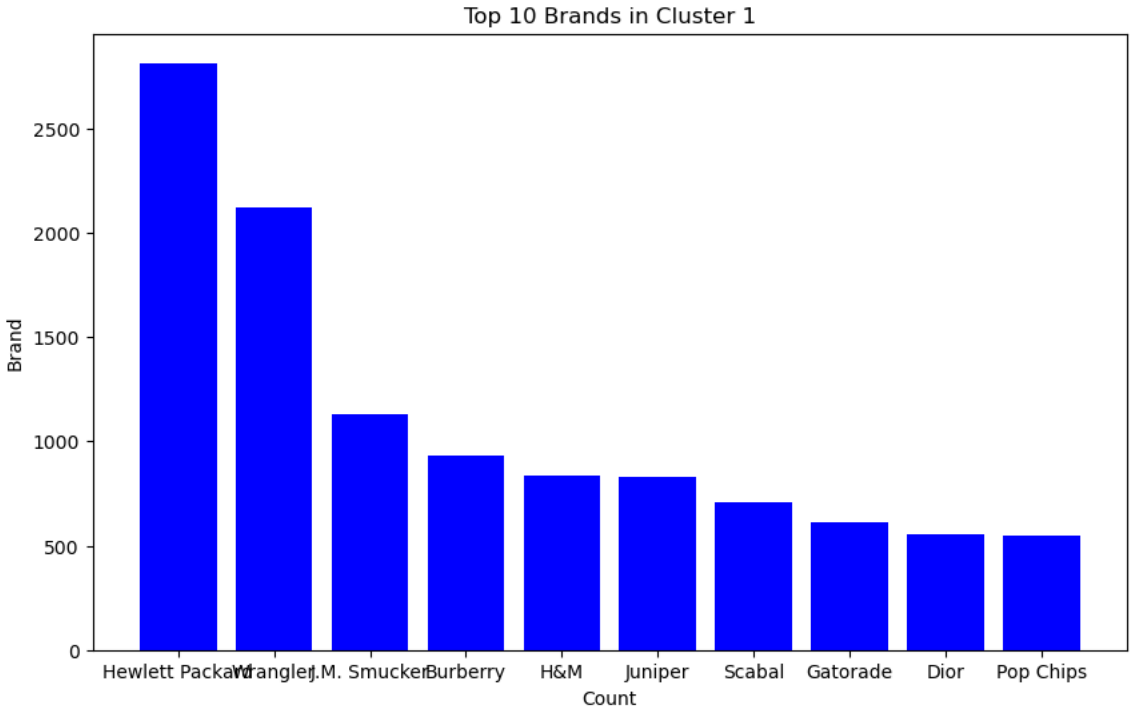
Out[43]:

	Brand	count
0	Hewlett Packard	2814
1	Wrangler	2120
2	J.M. Smucker	1132
3	Burberry	934
4	H&M	836
5	Juniper	830
6	Scabal	707
7	Gatorade	613
8	Dior	556
9	Pop Chips	550
10	Jordan	548
11	Dairy Queen	492
12	Huawei	445
13	Asics	431
14	Samsung	395
15	Fendi	389
16	LG	326
17	Asus	300
18	Colavita	287
19	Udis	280
20	Tommy Hilfiger	269
21	Mondelez International	262
22	Pladis	230
23	Vans	226
24	Hollister	223
25	Brooks	219
26	Mi	186
27	Microsoft	182
28	Jiffy mix	174
29	Compaq	172
30	Forever 21	170
31	Fila	121
32	Kraft	103
33	Siemens	39
34	IBM	34

Top 10 Brands in Cluster 1

In [44]:
```
top_10_cluster_1 = cluster_1_df.head(10)
```

In [45]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_10_cluster_1['Brand'], top_10_cluster_1['count'], color='blue')
plt.xlabel('Count')
plt.ylabel('Brand')
plt.title('Top 10 Brands in Cluster 1')
plt.show()
```



Cluster 1: Hewlett Packard, Wrangler, J.M. Smucker, Burberry, H&M, Juniper, Scabel, Gatorade, Dior, Pop chips

8.2 Cluster 2 Analysis

In [46]:
```
# Filter the data to select all rows belonging to Cluster 2
cluster_2 = final_cluster_data[final_cluster_data['Cluster labels'] == 1]
```

In [47]:
```
# Calculate and display the count of brands in Cluster 2
cluster_2_df = cluster_2.iloc[:,3:-1].sum().sort_values(ascending = False).reset_index()
cluster_2_df.columns = ['Brand', 'count']
cluster_2_df
```

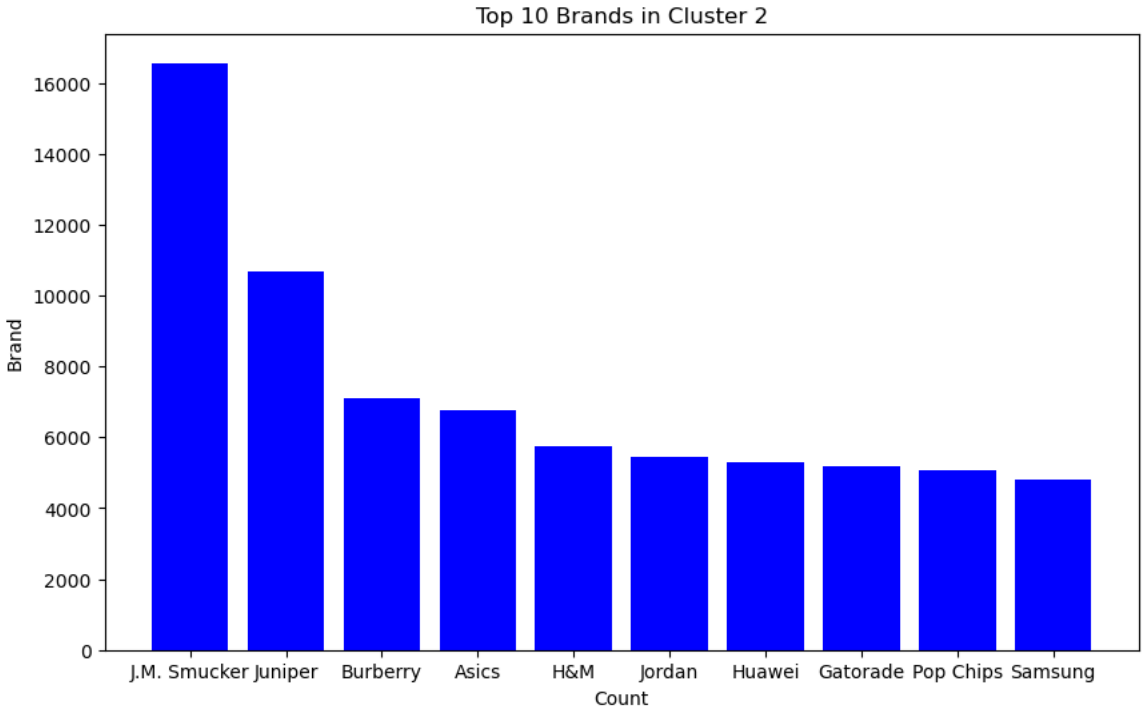
Out[47]:

	Brand	count
0	J.M. Smucker	16568
1	Juniper	10668
2	Burberry	7110
3	Asics	6775
4	H&M	5757
5	Jordan	5449
6	Huawei	5315
7	Gatorade	5199
8	Pop Chips	5072
9	Samsung	4810
10	Colavita	4176
11	Dior	4000
12	Dairy Queen	3752
13	Scabal	3622
14	Asus	3247
15	Brooks	2996
16	Fendi	2904
17	Compaq	2854
18	Udis	2804
19	Mondelez International	2759
20	Microsoft	2534
21	Vans	2503
22	Fila	2120
23	Mi	1984
24	LG	1840
25	Pladis	1643
26	Jiffy mix	1615
27	Kraft	1395
28	Hewlett Packard	1375
29	Siemens	1204
30	Tommy Hilfiger	1113
31	Hollister	1024
32	IBM	827
33	Forever 21	766
34	Wrangler	683

Top 10 Brands in Cluster 2

In [48]:
```
top_10_cluster_2 = cluster_2_df.head(10)
```

In [49]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_10_cluster_2['Brand'], top_10_cluster_2['count'], color='blue')
plt.xlabel('Count')
plt.ylabel('Brand')
plt.title('Top 10 Brands in Cluster 2')
plt.show()
```



Cluster 2: J.M. Smucker, Juniper, Burberry, Asics, H&M, Jordan, Huawei, Gatorade, Pop Chips, Samsung

8.3 Cluster 3 Analysis

In [50]:
```
# Filter the data to select all rows belonging to Cluster 3
cluster_3 = final_cluster_data[final_cluster_data['Cluster labels'] == 2]
```

In [51]:
```
# Calculate and display the count of brands in Cluster 3
cluster_3_df = cluster_3.iloc[:,3:-1].sum().sort_values(ascending = False).reset_index()
cluster_3_df.columns = ['Brand', 'count']
cluster_3_df
```

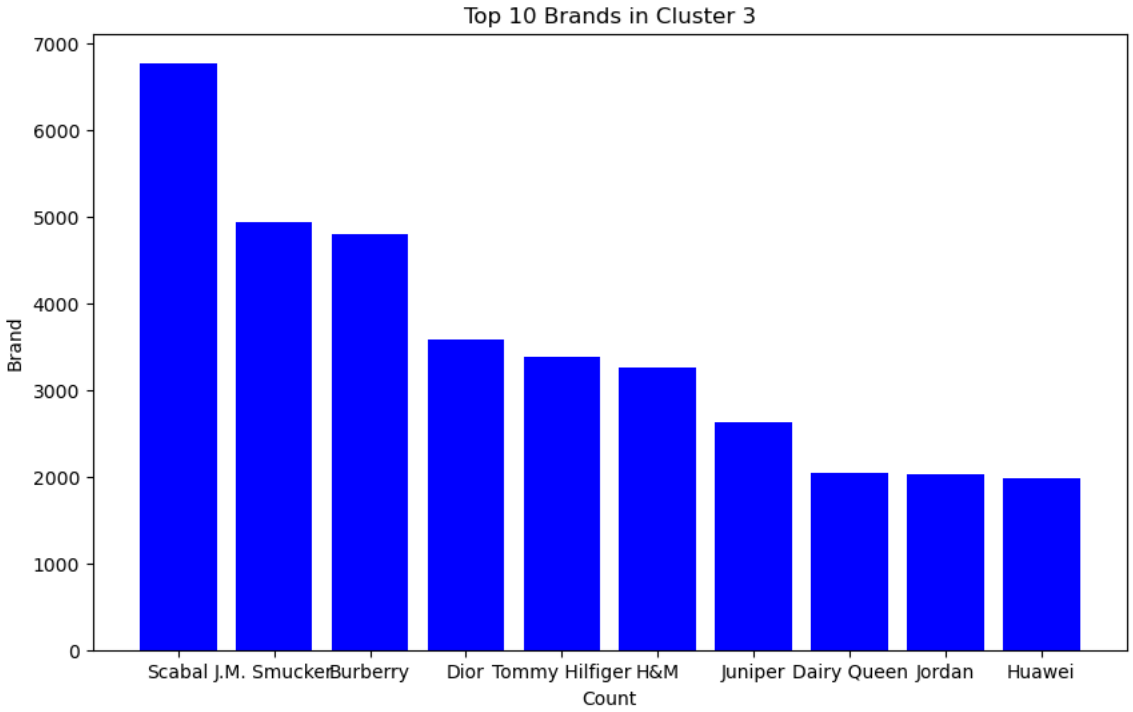
Out[51]:

	Brand	count
0	Scabal	6773
1	J.M. Smucker	4944
2	Burberry	4797
3	Dior	3578
4	Tommy Hilfiger	3387
5	H&M	3253
6	Juniper	2627
7	Dairy Queen	2036
8	Jordan	2026
9	Huawei	1980
10	Pop Chips	1815
11	Asics	1801
12	Gatorade	1758
13	Samsung	1483
14	Mi	1470
15	Pladis	1310
16	Colavita	1303
17	Asus	1293
18	Udis	1222
19	Mondelez International	1172
20	Hollister	1083
21	Brooks	989
22	Fendi	951
23	LG	910
24	Jiffy mix	852
25	Forever 21	784
26	Microsoft	775
27	Hewlett Packard	644
28	Kraft	629
29	Vans	614
30	Fila	587
31	Compaq	537
32	Wrangler	405
33	Siemens	199
34	IBM	98

Top 10 Brands in Cluster 3

In [52]:
```
top_10_cluster_3 = cluster_3_df.head(10)
```

In [53]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_10_cluster_3['Brand'], top_10_cluster_3['count'], color='blue')
plt.xlabel('Count')
plt.ylabel('Brand')
plt.title('Top 10 Brands in Cluster 3')
plt.show()
```



Cluster 3: Scabal, J. M. Smucker, Burberry, Dior, Tommy Hilfiger, H&M, Juniper, Dairy Queen, Jordan, Huawei

9. Conclusion

9.1 Cluster 1

1. This group of customers is primarily interested in brands such as Hewlett Packard, Wrangler, J.M. Smucker, Burberry, H&M, Juniper, Scabal, Gatorade, Dior, and Pop Chips. These brands may share similar customer demographics or preferences. Businesses can target this cluster with marketing strategies that focus on these brands.

1. Fashion and Clothing - **6** Food and Beverage - **3** Gadgets - **1**

Cluster 1

Brand	Industry	Specific
----------	----------	----------
Hewlett Packard	Gadgets	Information Technology
Wrangler	Fashion and Clothing	
J.M. Smucker	Food and Beverage	
Burberry	Fashion and Clothing	
H&M	Fashion and Clothing	
Juniper	Fashion and Clothing	Indian, Jaipur
Scabal	Fashion and Clothing	
Gatorade	Food and Beverage	Sport Themed
Dior	Fashion and Clothing	
Pop Chips	Food and Beverage	Also Tobacco

1. This Cluster has predominantly **Fashion and Clothing** , **Food and Beverage** brands in the Top 10.

9.2 Cluster 2

1. Customers in this cluster show an affinity for brands like J.M. Smucker, Juniper, Burberry, Asics, H&M, Jordan, Huawei, Gatorade, Pop Chips, and Samsung. This cluster might represent a different set of customer preferences compared to Cluster 1. Tailored marketing campaigns for these specific brands can be effective in engaging this segment.

1. Fashion and Clothing - **3** Food and Beverage - **3** Sport apparels - **2** Gadgets - **2**

Cluster 2

Brand	Industry	Specific
---	---	---
J.M. Smucker	Food and Beverage	
Juniper	Fashion and Clothing	Indian, Jaipur
Burberry	Fashion and Clothing	
Asics	Sport apparels	
H&M	Fashion and Clothing	
Jordan	Sport apparels	
Huawei	Gadgets	Networking Devices
Gatorade	Food and Beverage	Sport Themed
Pop chips	Food and Beverage	
Samsung	Gadgets	Electronics

1. This Cluster has a **mixer of all industry brands** in the Top 10.

9.3 Cluster 3

1. This cluster exhibits interest in brands such as Scabal, J.M. Smucker, Burberry, Dior, Tommy Hilfiger, H&M, Juniper, Dairy Queen, Jordan, and Huawei. It seems to have some overlap with Cluster 2 in terms of brand preferences. Marketing efforts could be designed to explore the connections between these brands and customer behaviors.

1. Fashion and Clothing - **6** Food and Beverage - **1** Sport apparels - **1** Gadgets - **1** Giftcards/Apparels - **1**

Cluster 3

Brand	Industry	Specific
---	---	---
Scabal	Fashion and Clothing	
J.M. Smucker	Food and Beverage	
Burberry	Fashion and Clothing	
Dior	Fashion and Clothing	
Tommy Hilfiger	Fashion and Clothing	
H&M	Fashion and Clothing	
Juniper	Fashion and Clothing	Indian, Jaipur
Dairy Queen	Gift Cards & Apparels	Restaurants
Jordan	Sport apparels	
Huawei	Gadgets	Networking Devices

1. This Cluster has a **Fashion and Clothing** in the Top 10.

Common brands in all the clusters

J. M . Smucker, Burberry, H&M, Juniper

Fashion& Clothing - **3** Food and Beverage -**1**

Common brands between cluster 1 & 2

J. M . Smucker, Burberry, H&M, Juniper, Gatorade, Pop chips

Fashion& Clothing - **3** Food and Beverage -**3**

Common brands between cluster 1 & 3

J. M . Smucker, Burberry, H&M, Juniper, Scabal, Dior

Fashion& Clothing - **5** Food and Beverage -**1**

Common brands between cluster 2 & 3

J. M . Smucker, Burberry, H&M, Juniper, Jordan, Huawei

Fashion& Clothing - **3** Food and Beverage -**1** Sports Apparels - **1** Gadgets - **1**

Project Insights and Implications

In summary, clustering analysis has helped us identify distinct groups of customers based on their brand preferences. By understanding these clusters, businesses can customize their marketing strategies to better target and engage with each group, potentially leading to improved customer satisfaction and increased sales. Further exploration and targeted campaigns can help uncover deeper insights into these customer segments.