In [33]:
```
import pandas as pd

# Load datasets
customers = pd.read_csv("Customers.csv")
products = pd.read_csv("Products.csv")
transactions = pd.read_csv("Transactions.csv")

# Display first few rows
print(customers.head())
print(products.head())
print(transactions.head())
```

```
  CustomerID        CustomerName         Region  SignupDate
0      C0001    Lawrence Carroll  South America  2022-07-10
1      C0002      Elizabeth Lutz           Asia  2022-02-13
2      C0003      Michael Rivera  South America  2024-03-07
3      C0004  Kathleen Rodriguez  South America  2022-10-09
4      C0005         Laura Weber           Asia  2022-08-15
  ProductID              ProductName     Category   Price
0      P001     ActiveWear Biography        Books  169.30
1      P002    ActiveWear Smartwatch  Electronics  346.30
2      P003  ComfortLiving Biography        Books   44.12
3      P004            BookWorld Rug   Home Decor   95.69
4      P005          TechPro T-Shirt     Clothing  429.31
  TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0        T00001      C0199      P067  2024-08-25 12:38:23         1
1        T00112      C0146      P067  2024-05-27 22:23:54         1
2        T00166      C0127      P067   2024-04-25 7:38:55         1
3        T00272      C0087      P067  2024-03-26 22:55:37         2
4        T00363      C0070      P067  2024-03-21 15:10:10         3

   TotalValue   Price
0      300.68  300.68
1      300.68  300.68
2      300.68  300.68
3      601.36  300.68
4      902.04  300.68
```

In [34]:
```
print(customers.isnull().sum())
print(products.isnull().sum())
print(transactions.isnull().sum())
```

```
CustomerID      0
CustomerName    0
Region          0
SignupDate      0
dtype: int64
ProductID      0
ProductName    0
Category       0
Price          0
dtype: int64
TransactionID      0
CustomerID         0
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price              0
dtype: int64
```

In [35]:
```
print(customers.duplicated().sum())
print(products.duplicated().sum())
print(transactions.duplicated().sum())
```

```
0
0
0
```

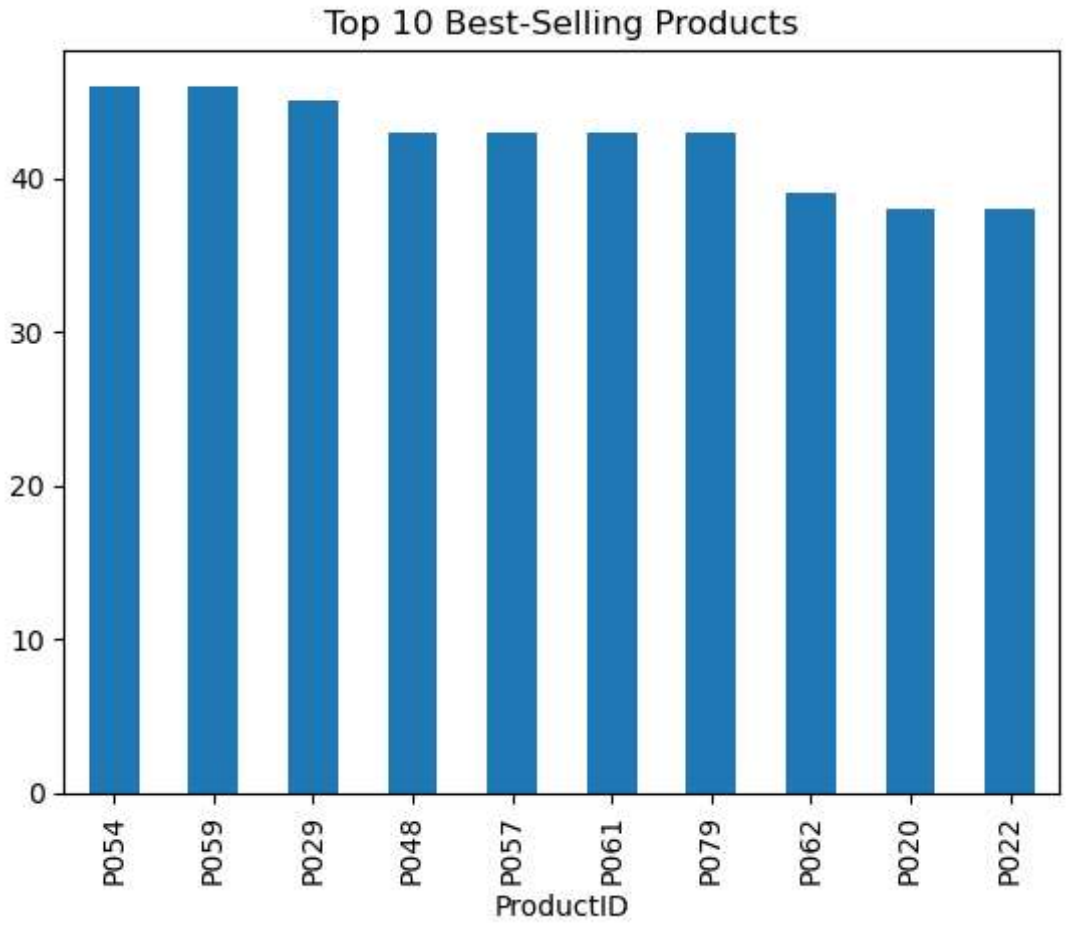
In [36]:
```
customers["SignupDate"] = pd.to_datetime(customers["SignupDate"])
transactions["TransactionDate"] = pd.to_datetime(transactions["TransactionDate"]
```

In [37]:
```
print(transactions.describe())
print(customers["Region"].value_counts())  # Check customer distribution by regi
```

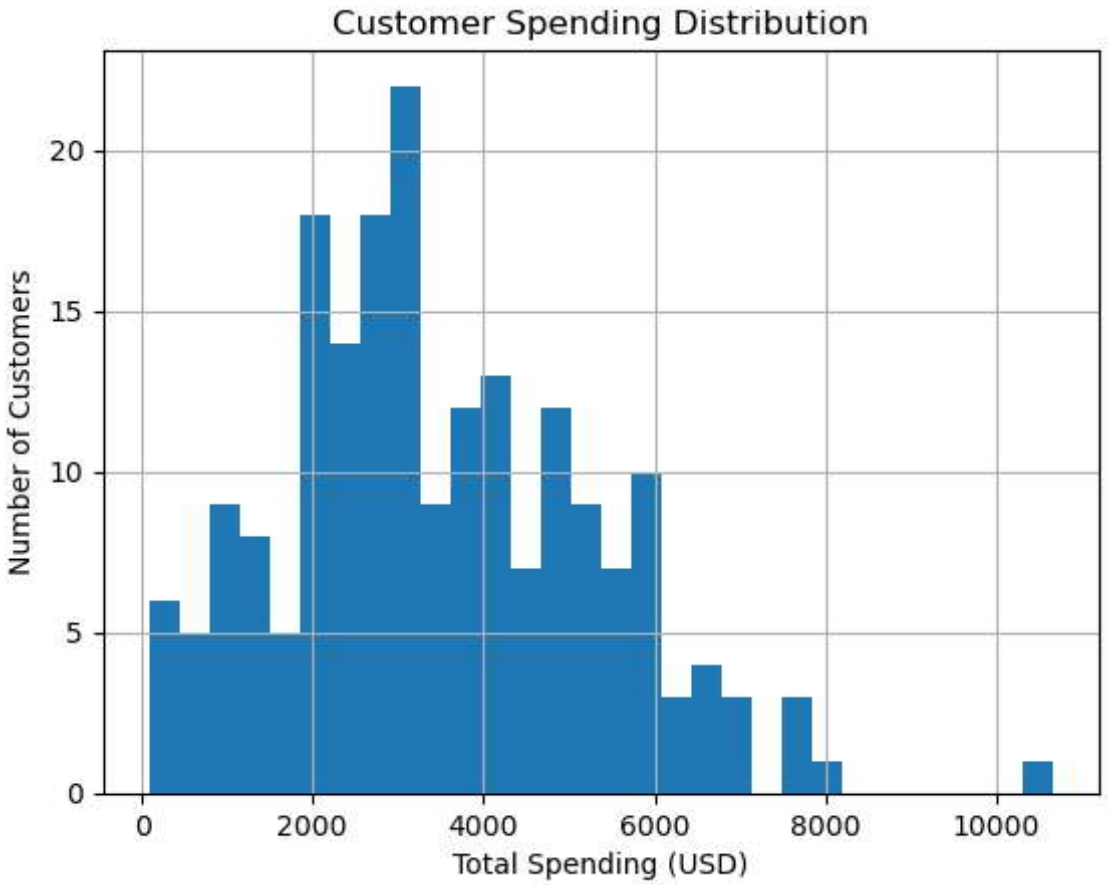
```
                     TransactionDate     Quantity   TotalValue       Price
count                           1000  1000.000000  1000.000000  1000.00000
mean   2024-06-23 15:33:02.768999936     2.537000   689.995560   272.55407
min              2023-12-30 15:29:12     1.000000    16.080000    16.08000
25%        2024-03-25 22:05:34.500000     2.000000   295.295000   147.95000
50%        2024-06-26 17:21:52.500000     3.000000   588.880000   299.93000
75%              2024-09-19 14:19:57     4.000000  1011.660000   404.40000
max              2024-12-28 11:00:00     4.000000  1991.040000   497.76000
std                              NaN     1.117981   493.144478   140.73639
Region
South America    59
Europe           50
North America    46
Asia             45
Name: count, dtype: int64
```

In [38]:
```
import matplotlib.pyplot as plt

top_products = transactions.groupby("ProductID")["Quantity"].sum().nlargest(10)
top_products.plot(kind="bar", title="Top 10 Best-Selling Products")
plt.show()
```



In [39]:
```
total_spent = transactions.groupby("CustomerID")["TotalValue"].sum()
total_spent.hist(bins=30)
plt.title("Customer Spending Distribution")
plt.xlabel("Total Spending (USD)")
plt.ylabel("Number of Customers")
plt.show()
```



In [40]:
```
customer_product_matrix = transactions.pivot_table(index="CustomerID", columns="
```

In [41]:
```
from sklearn.metrics.pairwise import cosine_similarity

similarity_matrix = cosine_similarity(customer_product_matrix)
similarity_df = pd.DataFrame(similarity_matrix, index=customer_product_matrix.ir
```

In [42]:
```
def get_similar_customers(customer_id, top_n=3):
    similar_customers = similarity_df[customer_id].sort_values(ascending=False)[
    return list(zip(similar_customers.index, similar_customers.values))

lookalike_results = {cust: get_similar_customers(cust) for cust in similarity_df
lookalike_df = pd.DataFrame(lookalike_results.items(), columns=["CustomerID", "S
lookalike_df.to_csv("YourName_Lookalike.csv", index=False)
```

In [43]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(customer_product_matrix)
```

In [44]:
```
# Merge customers with their transaction data
customer_transactions = transactions.groupby("CustomerID").agg(
    TotalSpent=("TotalValue", "sum"),
    PurchaseFrequency=("TransactionID", "count"),
).reset_index()

# Merge with the customers dataset
customers = customers.merge(customer_transactions, on="CustomerID", how="left").

# Select relevant features and scale
from sklearn.preprocessing import StandardScaler
```

```
features = customers[["TotalSpent", "PurchaseFrequency"]]
scaler = StandardScaler()
scaled_data = scaler.fit_transform(features)

# Apply KMeans
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4, random_state=42)
customers["Cluster"] = kmeans.fit_predict(scaled_data)
```

```
C:\Users\bandl\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [45]:
```
from sklearn.metrics import davies_bouldin_score

db_index = davies_bouldin_score(scaled_data, kmeans.labels_)
print(f"Davies-Bouldin Index: {db_index}")
```

```
Davies-Bouldin Index: 0.7963338823953641
```

In [ ]:

In [ ]:

In [ ]: