```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

transactions_dataset = pd.read_csv('Transactions.csv')
transactions_dataset['TransactionDate'] =
pd.to_datetime(transactions_dataset['TransactionDate'])
products_dataset = pd.read_csv('Products.csv')
customers_dataset = pd.read_csv('Customers.csv')

print(transactions_dataset)
# No of Numerical Features = 3
# No of Indexes = 3
# No of Datetime Features = 1
print(products_dataset)
# No of Numerical Features = 1
# No of Categorical Features = 1
print(customers_dataset)
# No of Datetime Features = 1
# No of Categorical Features = 1
```

```
    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 07:38:55         1

3          T00272      C0087      P067 2024-03-26 22:55:37         2

4          T00363      C0070      P067 2024-03-21 15:10:10         3

..            ...        ...       ...                 ...       ...

995        T00496      C0118      P037 2024-10-24 08:30:27         1

996        T00759      C0059      P037 2024-06-04 02:15:24         3

997        T00922      C0018      P037 2024-04-05 13:05:32         4

998        T00959      C0115      P037 2024-09-29 10:16:02         2

999        T00992      C0024      P037 2024-04-21 10:52:24         1


     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
..          ...     ...
995      459.86  459.86
996     1379.58  459.86
997     1839.44  459.86
998      919.72  459.86
999      459.86  459.86

[1000 rows x 7 columns]
   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
..       ...                      ...          ...     ...
95      P096     SoundWave Headphones  Electronics  307.47
96      P097       BookWorld Cookbook        Books  319.34
97      P098         SoundWave Laptop  Electronics  299.93
98      P099   SoundWave Mystery Book        Books  354.29
99      P100         HomeSense Sweater     Clothing  126.34

[100 rows x 4 columns]
    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
..         ...                 ...            ...         ...
195      C0196         Laura Watts         Europe  2022-06-07
196      C0197    Christina Harvey         Europe  2023-03-21
197      C0198         Rebecca Ray         Europe  2022-02-27
198      C0199      Andrea Jenkins         Europe  2022-12-03
199      C0200         Kelly Cross           Asia  2023-06-11

[200 rows x 4 columns]
```

```
print(f" Null values in Transactions Dataset: \n
{transactions_dataset.isnull().sum()}\n\n")
print(f" Null values in Products Dataset:\n
{products_dataset.isnull().sum()}\n\n")
print(f" Null values in Customers Dataset: \
n{customers_dataset.isnull().sum()}")
#No null values
```

```
 Null values in Transactions Dataset:
 TransactionID      0
CustomerID         0
```

```
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price              0
dtype: int64


 Null values in Products Dataset:
 ProductID      0
ProductName    0
Category       0
Price          0
dtype: int64


 Null values in Customers Dataset:
CustomerID      0
CustomerName    0
Region          0
SignupDate      0
dtype: int64
```

```
print(transactions_dataset.columns)
print(products_dataset.columns)
print(customers_dataset.columns)
```

```
Index(['TransactionID', 'CustomerID', 'ProductID', 'TransactionDate',
       'Quantity', 'TotalValue', 'Price'],
      dtype='object')
Index(['ProductID', 'ProductName', 'Category', 'Price'],
dtype='object')
Index(['CustomerID', 'CustomerName', 'Region', 'SignupDate'],
dtype='object')
```

```
print(transactions_dataset.info())
print(products_dataset.info())
print(customers_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    1000 non-null   object
 1   CustomerID       1000 non-null   object
 2   ProductID        1000 non-null   object
 3   TransactionDate  1000 non-null   datetime64[ns]
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
```

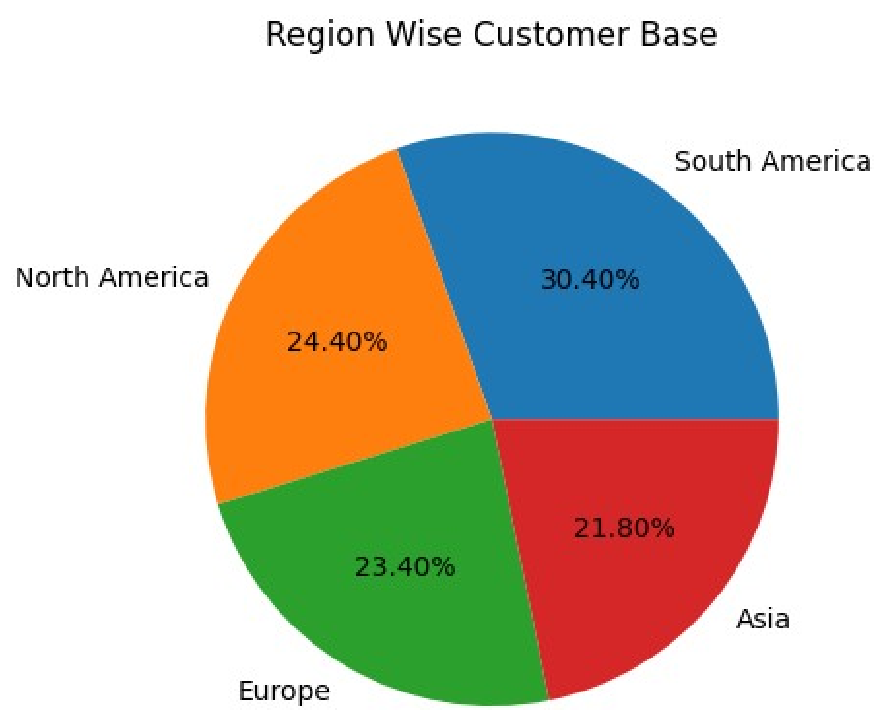
```
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 54.8+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ProductID    100 non-null    object
 1   ProductName  100 non-null    object
 2   Category     100 non-null    object
 3   Price        100 non-null    float64
dtypes: float64(1), object(3)
memory usage: 3.3+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
None

dataset = transactions_dataset.merge(products_dataset,on =
"ProductID",how = 'left').merge(customers_dataset,on =
"CustomerID",how = 'left')
print(dataset)

    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 07:38:55         1

3          T00272      C0087      P067 2024-03-26 22:55:37         2

4          T00363      C0070      P067 2024-03-21 15:10:10         3

..            ...        ...       ...                 ...       ...

995        T00496      C0118      P037 2024-10-24 08:30:27         1

996        T00759      C0059      P037 2024-06-04 02:15:24         3
```

```
997        T00922      C0018      P037 2024-04-05 13:05:32         4
998        T00959      C0115      P037 2024-09-29 10:16:02         2
999        T00992      C0024      P037 2024-04-21 10:52:24         1

     TotalValue  Price_x                      ProductName     Category  \
0        300.68   300.68  ComfortLiving Bluetooth Speaker  Electronics
1        300.68   300.68  ComfortLiving Bluetooth Speaker  Electronics
2        300.68   300.68  ComfortLiving Bluetooth Speaker  Electronics
3        601.36   300.68  ComfortLiving Bluetooth Speaker  Electronics
4        902.04   300.68  ComfortLiving Bluetooth Speaker  Electronics
..          ...      ...                              ...          ...
995      459.86   459.86            SoundWave Smartwatch  Electronics
996     1379.58   459.86            SoundWave Smartwatch  Electronics
997     1839.44   459.86            SoundWave Smartwatch  Electronics
998      919.72   459.86            SoundWave Smartwatch  Electronics
999      459.86   459.86            SoundWave Smartwatch  Electronics

     Price_y          CustomerName         Region  SignupDate
0     300.68        Andrea Jenkins         Europe  2022-12-03
1     300.68       Brittany Harvey           Asia  2024-09-04
2     300.68       Kathryn Stevens         Europe  2024-04-04
3     300.68       Travis Campbell  South America  2024-04-11
4     300.68         Timothy Perez         Europe  2022-03-15
..       ...                   ...            ...         ...
995   459.86            Jacob Holt  South America  2022-01-22
996   459.86  Mrs. Kimberly Wright  North America  2024-04-07
997   459.86          Tyler Haynes  North America  2024-09-21
998   459.86       Joshua Hamilton           Asia  2024-11-11
999   459.86         Michele Cooley  North America  2024-02-05

[1000 rows x 13 columns]
```

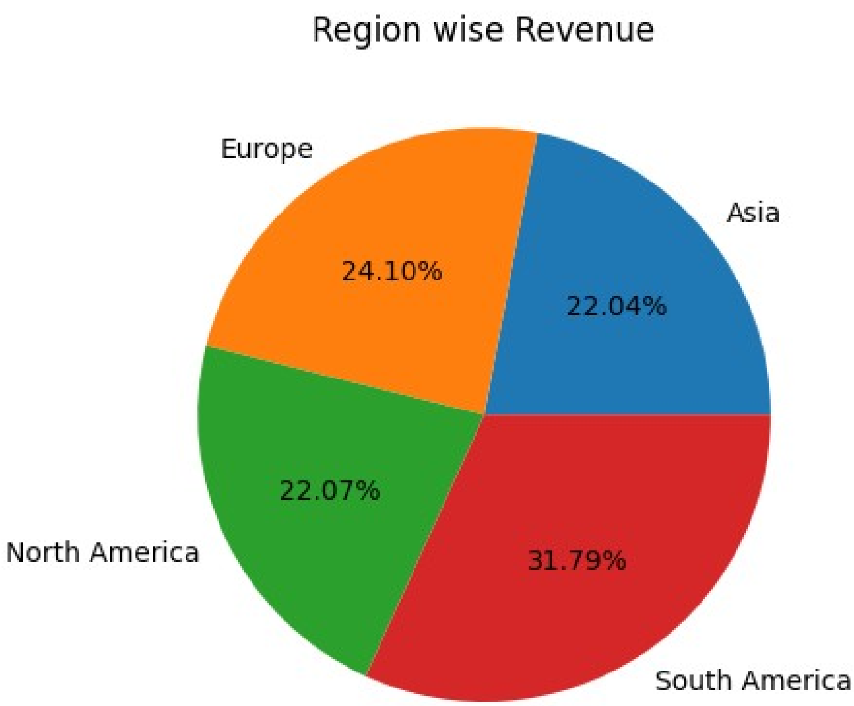
```
print(dataset.isnull().sum())
```

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

```
regions = dataset.Region.value_counts().index
regions_values = dataset.Region.value_counts().values
plt.pie(regions_values,labels=regions,autopct='%1.2f%%')
plt.title('Region Wise Customer Base')
plt.show()

regional_revenue = dataset.groupby(['Region']).apply(lambda x:
(x['Quantity']*x['Price_x']).sum()).to_dict()
plt.pie(regional_revenue.values(),labels=regional_revenue.keys(),autop
ct='%1.2f%%')
plt.title('Region wise Revenue')
plt.show()
```



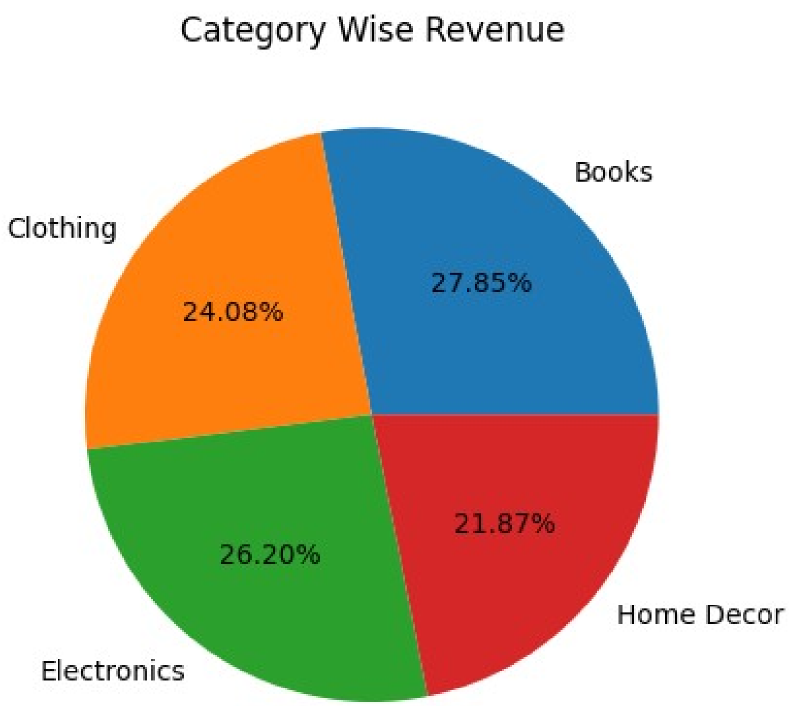

```
<ipython-input-29-5d0788cdaa6a>:7: DeprecationWarning:
DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  regional_revenue = dataset.groupby(['Region']).apply(lambda x:
(x['Quantity']*x['Price_x']).sum()).to_dict()
```



South America has the largest Customer Base followed by North America, Europe and Asia in order.

South America also has the Largest Revenue Contribution Region wise,followed by Europe, North America, Asia respectively.

```
categories = dataset.Category.value_counts().index
categories_values = dataset.Category.value_counts().values
cat_revenue = dataset.groupby(['Category'])
['TotalValue'].sum().reset_index()
print(cat_revenue)
plt.pie(cat_revenue['TotalValue'],labels=cat_revenue['Category'],autop
ct='%1.2f%%')
plt.title('Category Wise Revenue')
plt.show()

      Category  TotalValue
0        Books   192147.47
1     Clothing   166170.66
2  Electronics   180783.50
3   Home Decor   150893.93
```



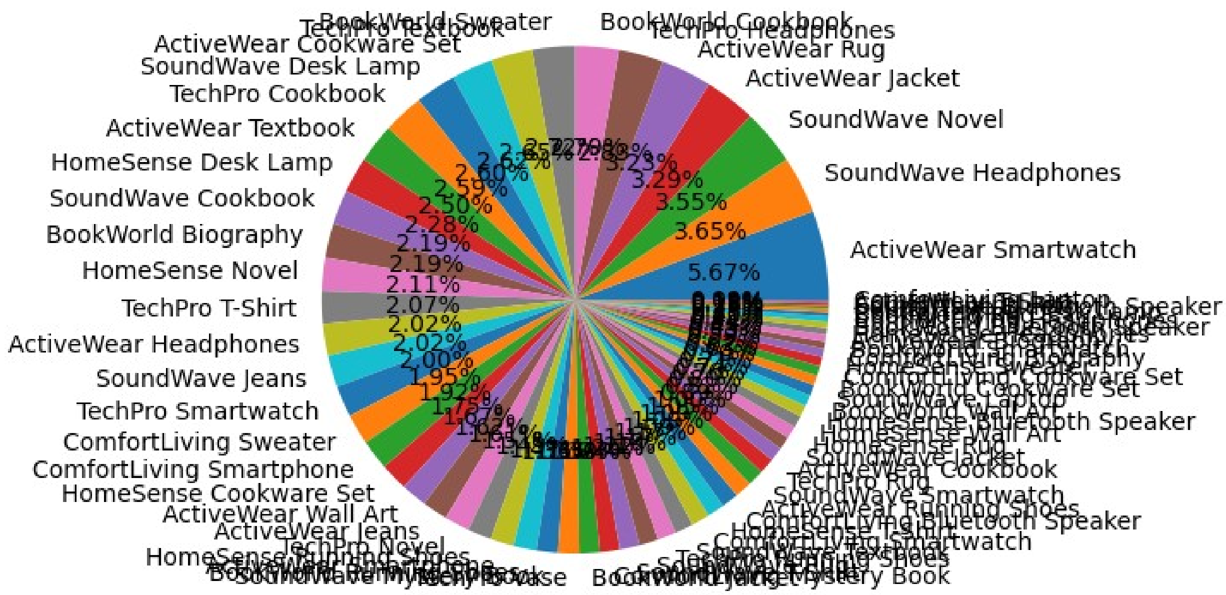
Biggest Product Categories are Books followed by Electronics,Home Decor and Clothing in order

```
# quantity = dataset.Quantity
# product_names = dataset.ProductName
# prices = dataset.Price_x
# resullt = {}
# for j in range(len(product_names)):
#   if product_names[j] in resullt.keys():
#     resullt[product_names[j]] += quantity[j]*prices[j]
#   else:
#     resullt[product_names[j]] = quantity[j]*prices[j]

result = (dataset.groupby('ProductName').apply(lambda x:
(x['Quantity'] * x['Price_x']).sum()).to_dict())

result = dict(sorted(result.items(), key=lambda x: x[1],
reverse=True))
print(result)

plt.pie(list(result.values()),labels=list(result.keys()),autopct='%1.2
f%%')
plt.show()
print("Top 5 Most Revenue Generating Products")
plt.pie(list(result.values())[:5],labels=list(result.keys())
[:5],autopct='%1.2f%%')
```

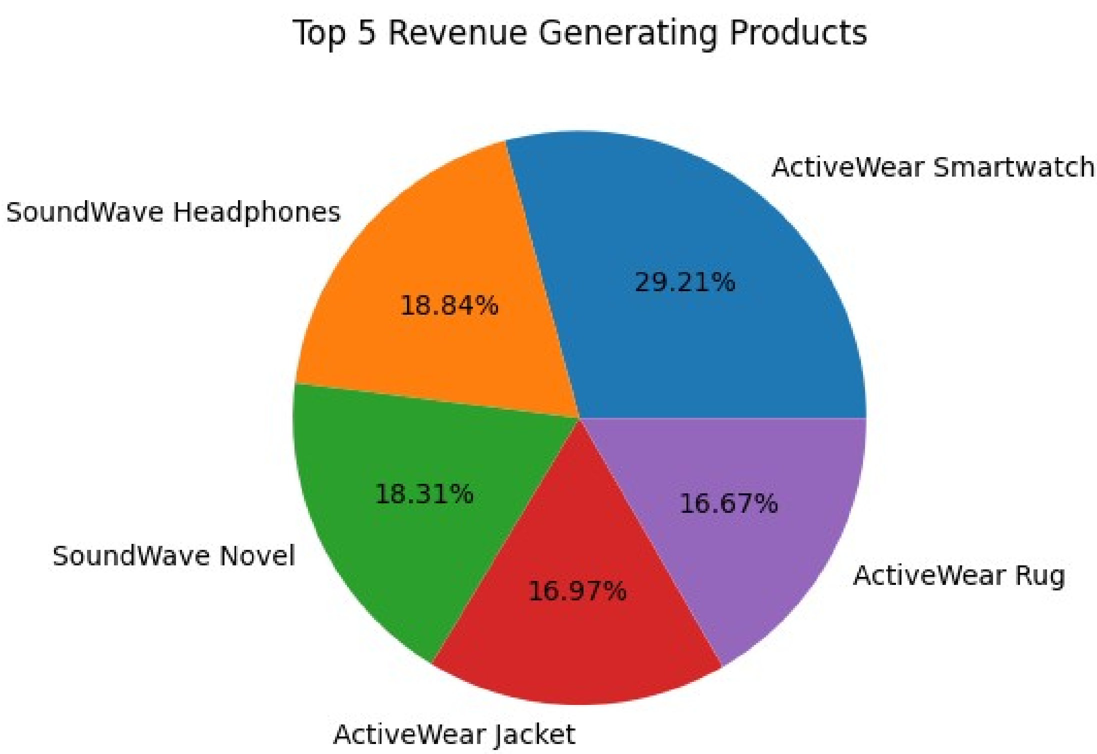
```python
plt.title('Top 5 Revenue Generating Products')
plt.show()
```

```
<ipython-input-30-728fe8ae9678>:11: DeprecationWarning:
DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  result = (dataset.groupby('ProductName').apply(lambda x:
(x['Quantity'] * x['Price_x']).sum()).to_dict())
```


```
{'ActiveWear Smartwatch': 39096.97, 'SoundWave Headphones':
25211.640000000003, 'SoundWave Novel': 24507.899999999998, 'ActiveWear
Jacket': 22712.559999999998, 'ActiveWear Rug': 22314.43, 'TechPro
Headphones': 19513.8, 'BookWorld Cookbook': 19221.99, 'BookWorld
Sweater': 18743.789999999997, 'TechPro Textbook': 18267.96,
'ActiveWear Cookware Set': 18083.730000000003, 'SoundWave Desk Lamp':
17920.100000000002, 'TechPro Cookbook': 17905.2, 'ActiveWear
Textbook': 17257.860000000004, 'HomeSense Desk Lamp':
15701.319999999998, 'SoundWave Cookbook': 15102.719999999998,
'BookWorld Biography': 15080.210000000003, 'HomeSense Novel':
14592.240000000002, 'TechPro T-Shirt': 14264.14, 'ActiveWear
Headphones': 13958.34, 'SoundWave Jeans': 13947.199999999999, 'TechPro
Smartwatch': 13778.880000000001, 'ComfortLiving Sweater': 13487.95,
'ComfortLiving Smartphone': 13232.12, 'HomeSense Cookware Set':
12078.18, 'ActiveWear Wall Art': 11488.94, 'ActiveWear Jeans':
11161.54, 'TechPro Novel': 11126.040000000003, 'HomeSense Running
Shoes': 10405.66, 'ActiveWear Smartphone': 10307.1, 'BookWorld Running
Shoes': 10119.2, 'SoundWave Mystery Book': 9412.03, 'TechPro Vase':
9306.14, 'BookWorld Jacket': 8941.199999999999, 'ComfortLiving Mystery
Book': 8737.8, 'SoundWave T-Shirt': 8672.039999999999, 'SoundWave
Rug': 8396.0, 'TechPro Running Shoes': 8124.76, 'SoundWave Textbook':
8093.280000000001, 'ComfortLiving Smartwatch': 8052.99, 'HomeSense T-
Shirt': 7528.9, 'ComfortLiving Bluetooth Speaker': 7516.999999999999,
'ActiveWear Running Shoes': 7505.759999999999, 'SoundWave Smartwatch':
7235.58, 'TechPro Rug': 6873.5, 'ActiveWear Cookbook': 6112.92,
'SoundWave Jacket': 5676.96, 'HomeSense Rug': 5529.68, 'HomeSense Wall
Art': 5226.5199999999995, 'HomeSense Bluetooth Speaker': 5080.93,
'BookWorld Wall Art': 4875.15, 'SoundWave Laptop': 4798.880000000001,
'BookWorld Cookware Set': 4317.94, 'ComfortLiving Cookware Set':
4301.91, 'HomeSense Sweater': 4083.66, 'ComfortLiving Biography':
3681.92, 'BookWorld Smartwatch': 3083.4, 'ActiveWear Biography':
3047.4, 'HomeSense Headphones': 2860.7400000000002, 'BookWorld
Bluetooth Speaker': 2790.1499999999996, 'ComfortLiving Headphones':
2394.0, 'BookWorld Rug': 1722.4199999999998, 'ComfortLiving Desk
Lamp': 1694.1599999999999, 'SoundWave Bluetooth Speaker':
1223.2199999999998, 'ComfortLiving Rug': 1063.81, 'ActiveWear T-
Shirt': 795.34, 'ComfortLiving Laptop': 647.76}
```

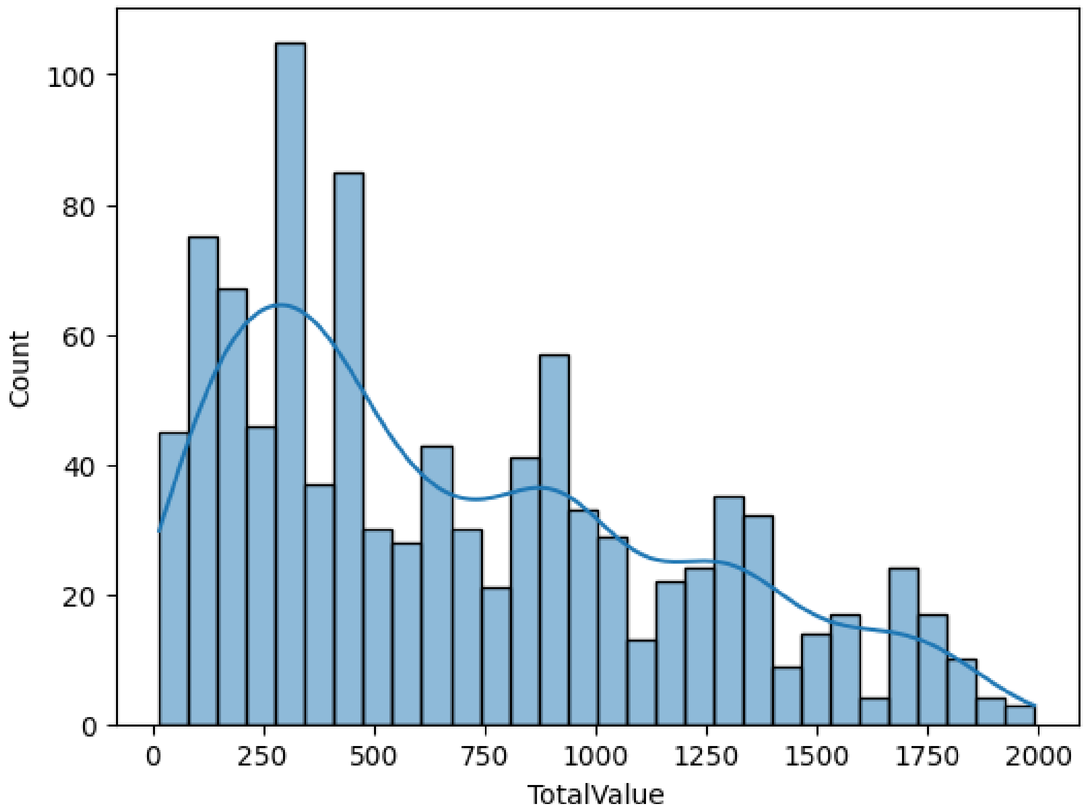


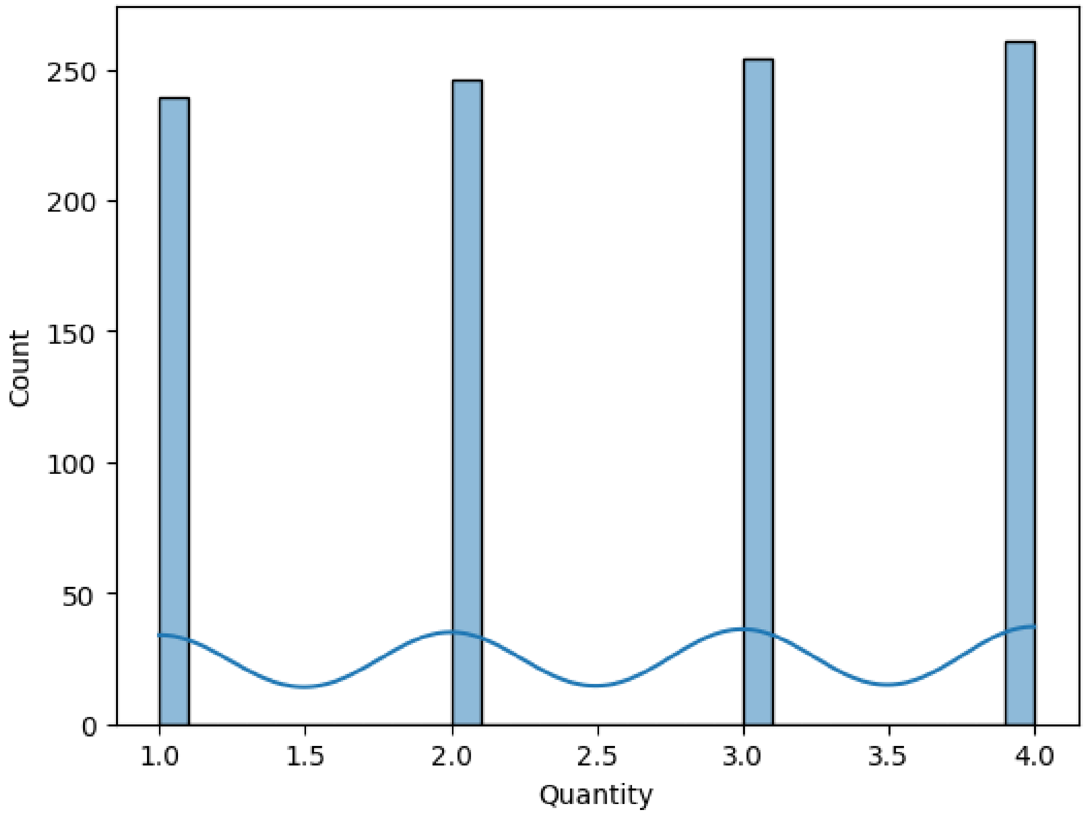

```
Top 5 Most Revenue Generating Products
```



ActiveWear Smartwatch is the most Revenue Generating Product contributing to 5.67% of Total Revenue

```
sns.histplot(dataset['TotalValue'],bins = 30,kde=True)
plt.show()
sns.histplot(dataset['Quantity'],bins = 30,kde=True)
plt.show()
```





Majority of Orders are in 0-500$ range and the trend goes down as the Transaction Value increases.

Customers also prefer to buy in bulk.

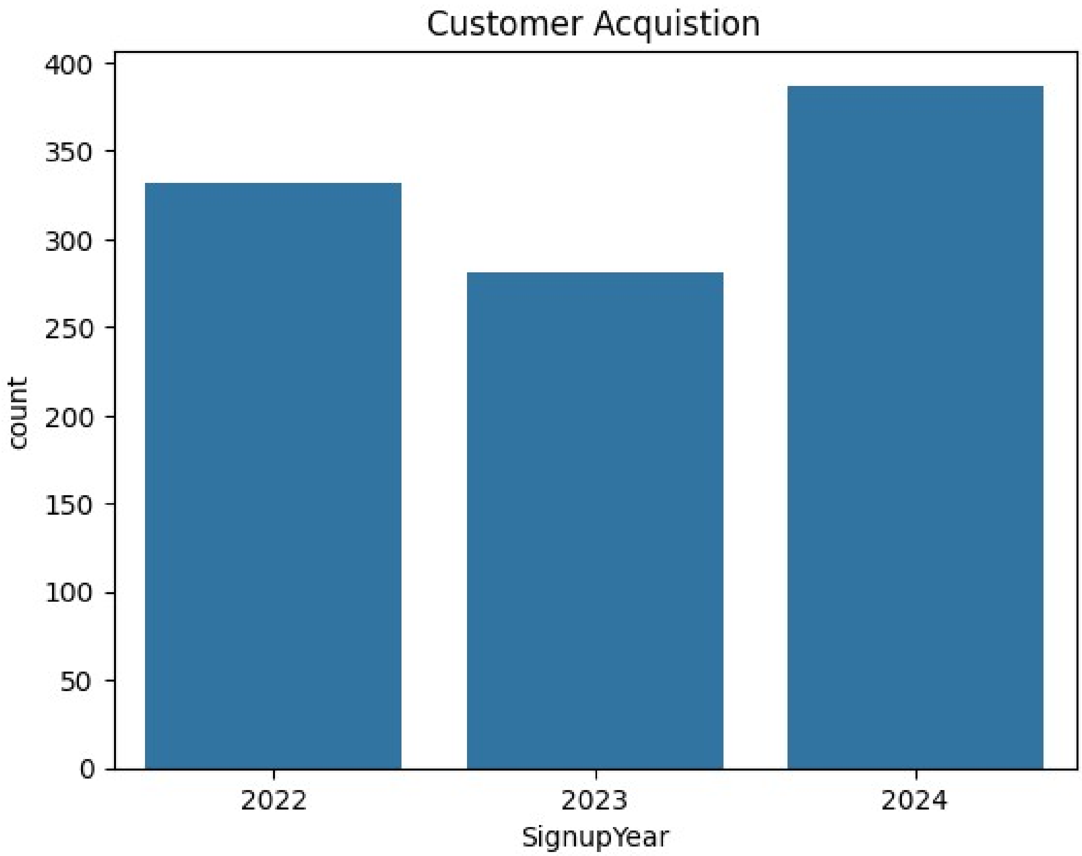
```
correlation_matrix = dataset[['Quantity', 'Price_x',
'TotalValue']].corr()
print(correlation_matrix)
```

```
             Quantity   Price_x  TotalValue
Quantity     1.000000 -0.009378    0.609972
Price_x     -0.009378  1.000000    0.722714
TotalValue   0.609972  0.722714    1.000000
```

```
dataset['SignupDate'] = pd.to_datetime(dataset['SignupDate'])
dataset['SignupYear'] = dataset['SignupDate'].dt.year
values = dataset['SignupYear'].value_counts().sort_index()
growth = values.pct_change()*100
print(growth)
sns.countplot(x='SignupYear',data=dataset)
plt.title('Customer Acquistion')
plt.show()
```

```
SignupYear
2022              NaN
```

```
2023   -15.361446
2024    37.722420
Name: count, dtype: float64
```

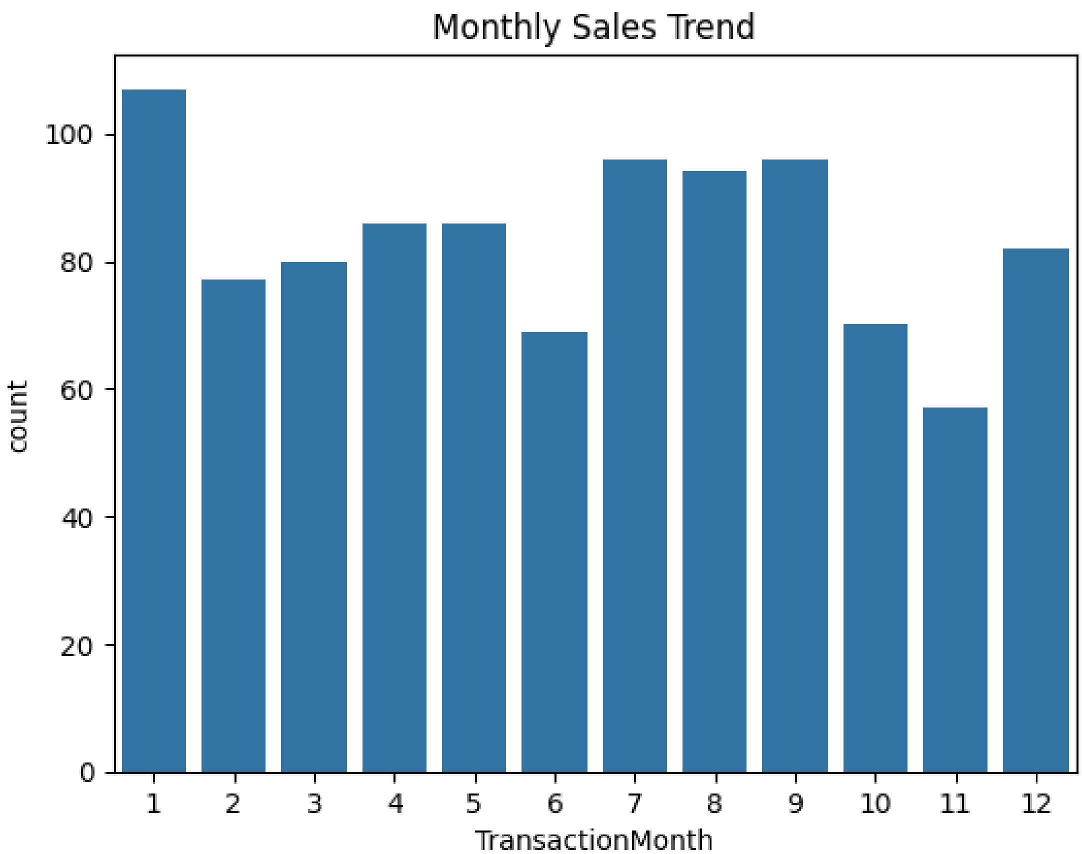


Customer Acquisition has gone up by 37.7% in the year 2024, but the year 2023 saw a decline of 15%

```
dataset['TransactionDate'] =
pd.to_datetime(dataset['TransactionDate'])
dataset['TransactionMonth'] = dataset['TransactionDate'].dt.month
values = dataset['TransactionMonth'].value_counts().sort_index()
growth = values.pct_change()*100
print(growth)
sns.countplot(x='TransactionMonth',data = dataset)
plt.title('Monthly Sales Trend')
plt.show()
```

```
TransactionMonth
1           NaN
2    -28.037383
3      3.896104
4      7.500000
5      0.000000
```

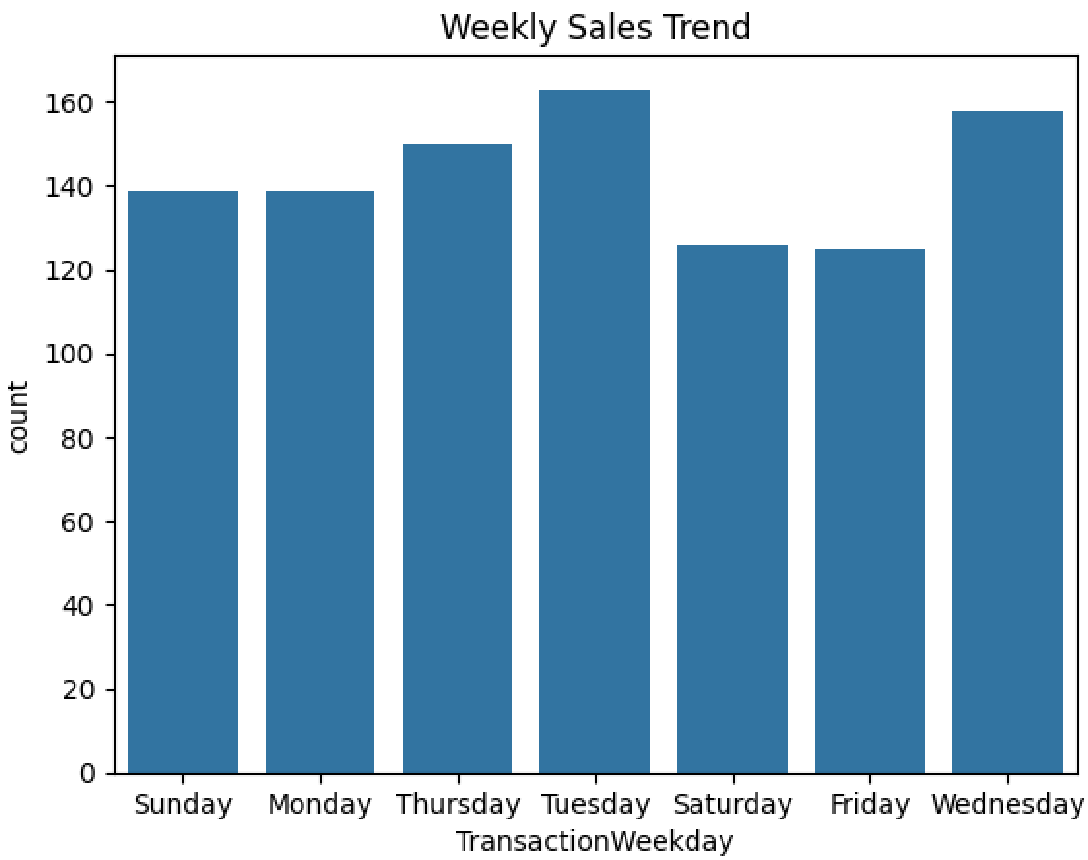
```
6    -19.767442
7     39.130435
8     -2.083333
9      2.127660
10   -27.083333
11   -18.571429
12    43.859649
Name: count, dtype: float64
```



Highest Sales is seen in January probably due to New Year.July to Sepetember is also a great period for sales while November is the worst month for sales in the Year.|

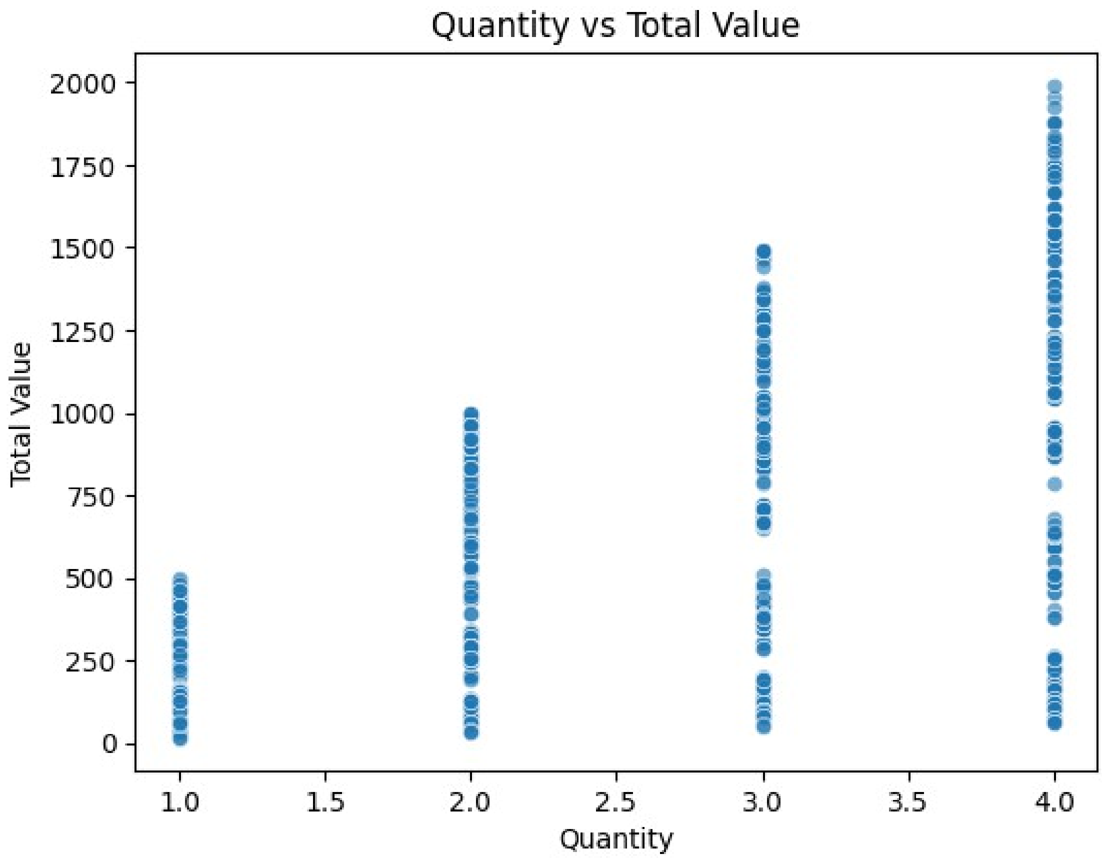
```
dataset['TransactionDate'] =
pd.to_datetime(dataset['TransactionDate'])
dataset['TransactionWeekday'] =
dataset['TransactionDate'].dt.day_name()
values = dataset['TransactionWeekday'].value_counts().sort_index()
growth = values.pct_change()*100
print(growth)
sns.countplot(x='TransactionWeekday',data = dataset)
plt.title('Weekly Sales Trend')
plt.show()
```

```
TransactionWeekday
Friday             NaN
Monday       11.200000
Saturday     -9.352518
Sunday       10.317460
Thursday      7.913669
Tuesday       8.666667
Wednesday    -3.067485
Name: count, dtype: float64
```



Weekends see relatively less Transactions, Whereas weekdays are quite good for Sales.

```
sns.scatterplot(x='Quantity', y='TotalValue', data=dataset, alpha=0.6)
plt.title('Quantity vs Total Value')
plt.xlabel('Quantity')
plt.ylabel('Total Value')
plt.show()
```




```
import datetime
last_purchase_customers = dataset.groupby('CustomerID')
['TransactionDate'].max().reset_index()
days_since_last_purchase = datetime.datetime.now() -
last_purchase_customers['TransactionDate']
last_purchase_customers['DaysSinceLastPurchase'] =
days_since_last_purchase.dt.days
last_purchase_customers =
last_purchase_customers.sort_values(by='DaysSinceLastPurchase')
last_purchase_customers
```

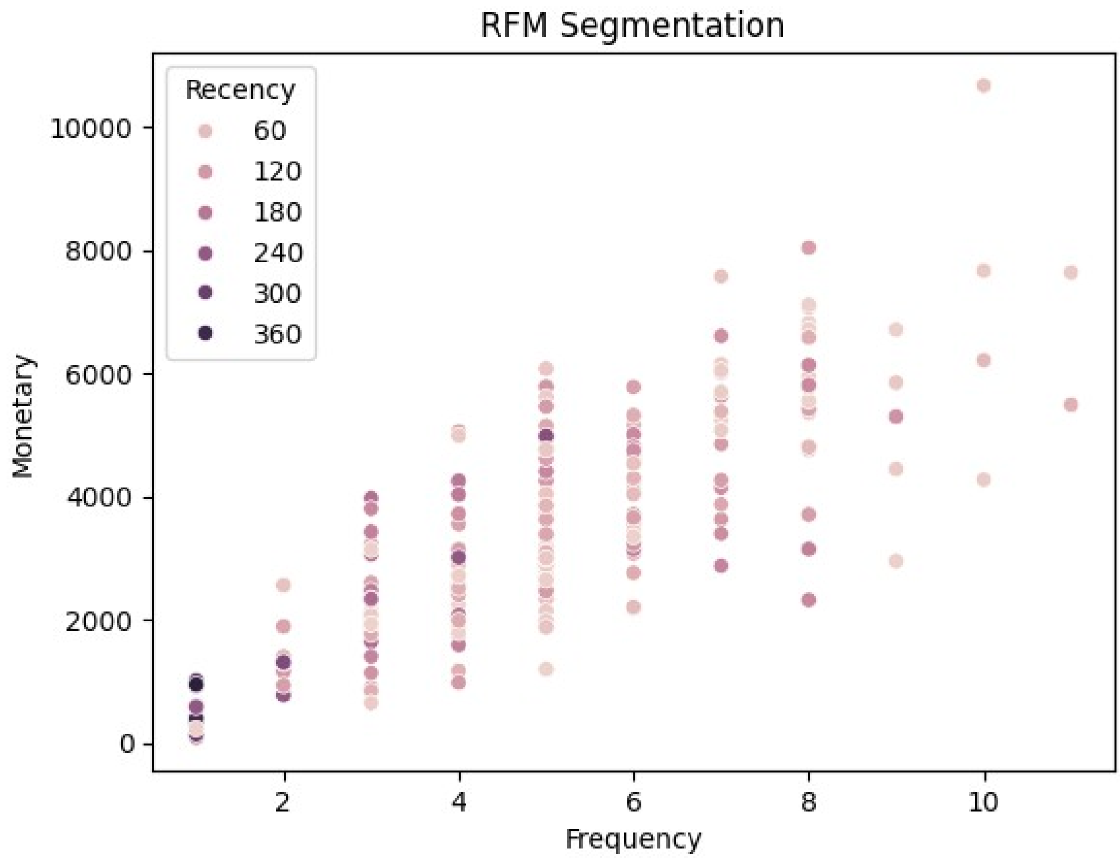
{"summary":"{\n \"name\": \"last_purchase_customers\",\n \"rows\": 199,\n \"fields\": [\n   {\n   \"column\": \"CustomerID\",\n \"properties\": {\n   \"dtype\": \"string\",\n \"num_unique_values\": 199,\n   \"samples\": [\n \"C0016\",\n   \"C0137\",\n   \"C0152\"\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"TransactionDate\",\n \"properties\": {\n   \"dtype\": \"date\",\n   \"min\": \"2024-01-02 19:11:34\",\n   \"max\": \"2024-12-28 11:00:00\",\n \"num_unique_values\": 199,\n   \"samples\": [\n   \"2024-11-10 18:24:04\",\n   \"2024-12-23 23:09:50\",\n \"2024-10-21 06:20:03\"\n   ],\n   \"semantic_type\": \"\",\

```
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"DaysSinceLastPurchase\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 70,\n        \"min\": 27,\n
\"max\": 387,\n        \"num_unique_values\": 121,\n
\"samples\": [\n          80,\n          83,\n          31\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"last_purchase_customers"}
```

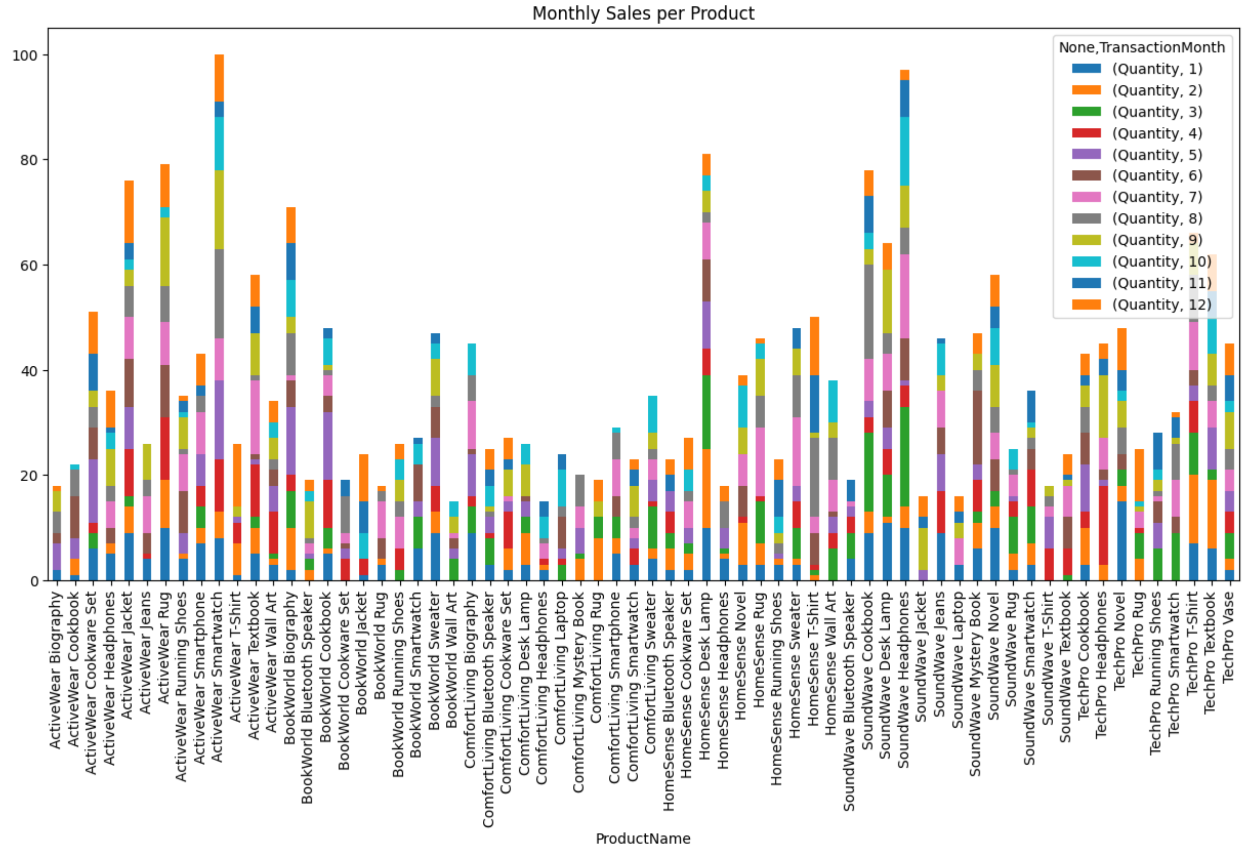

Some Customers have gone over a year since making a purchase we could Target them with a Coupon Code

```
rfm = dataset.groupby('CustomerID').agg({
    'TransactionDate': lambda x: (datetime.datetime.now() -
x.max()).days,
    'TransactionID': 'count',
    'TotalValue': 'sum'
}).rename(columns={
    'TransactionDate': 'Recency',
    'TransactionID': 'Frequency',
    'TotalValue': 'Monetary'
})
print(rfm.head())
sns.scatterplot(data=rfm, x='Frequency', y='Monetary', hue='Recency')
plt.title('RFM Segmentation')
plt.show()
```

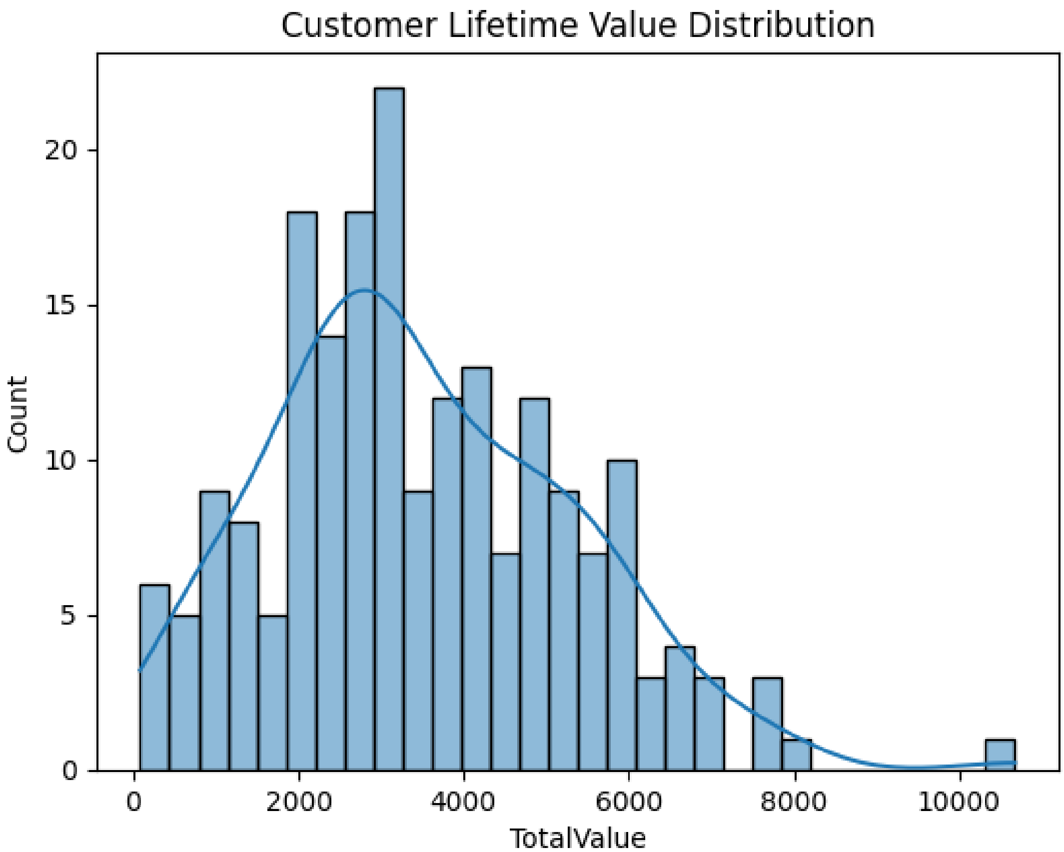
	Recency	Frequency	Monetary
CustomerID			
C0001	82	5	3354.52
C0002	52	4	1862.74
C0003	152	4	2725.38
C0004	32	8	5354.88
C0005	81	3	2034.24




```
product_sales = dataset.groupby(['ProductName',
'TransactionMonth']).agg({'Quantity': 'sum'}).unstack().fillna(0)
product_sales.plot(kind='bar', stacked=True, figsize=(15, 7))
plt.title('Monthly Sales per Product')
plt.show()
```



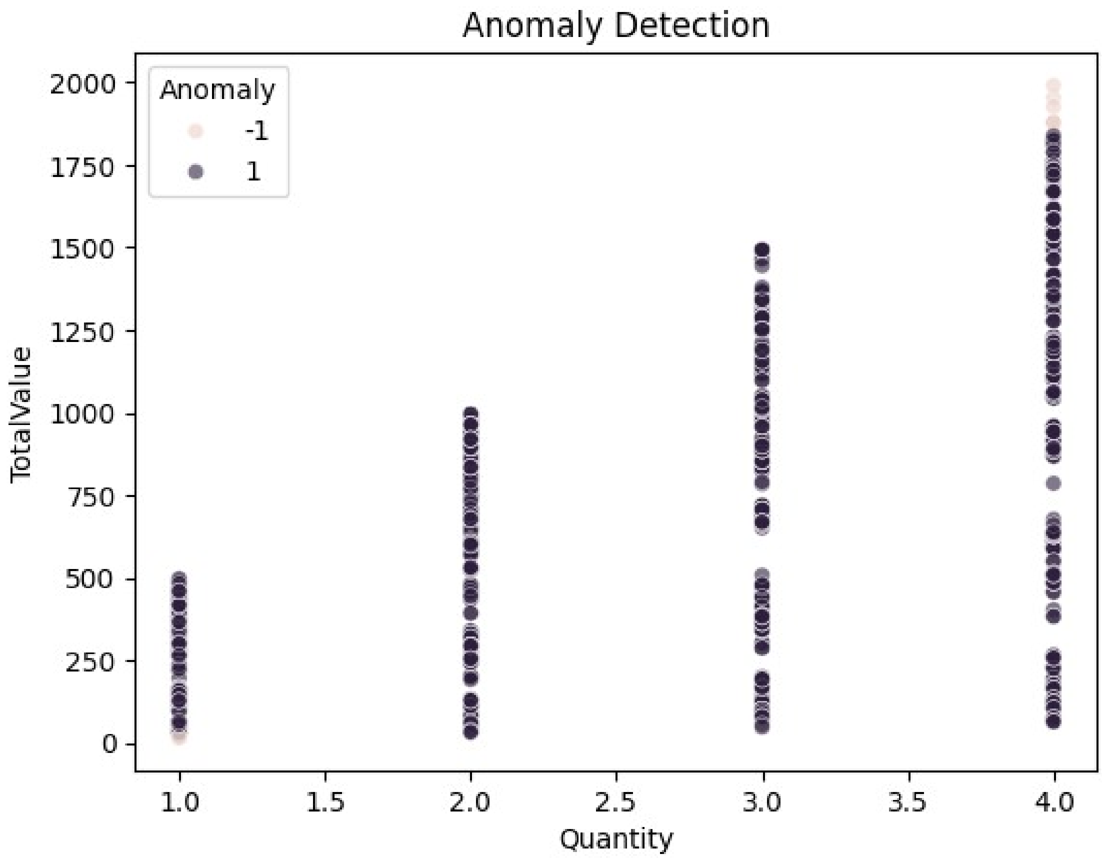

```
clv = dataset.groupby('CustomerID')
['TotalValue'].sum().sort_values(ascending=False)
sns.histplot(clv, bins=30, kde=True)
plt.title('Customer Lifetime Value Distribution')
plt.show()
```




```
repeat_rate =
dataset.groupby('CustomerID').size().value_counts(normalize=True).iloc
[1:].sum()
print(f"Repeat Purchase Rate: {repeat_rate * 100:.2f}%")
```

```
Repeat Purchase Rate: 78.89%
```

```
from sklearn.ensemble import IsolationForest

iso = IsolationForest(contamination=0.01, random_state=42)
dataset['Anomaly'] = iso.fit_predict(dataset[['TotalValue',
'Quantity']])
sns.scatterplot(data=dataset, x='Quantity', y='TotalValue',
hue='Anomaly', alpha=0.6)
plt.title('Anomaly Detection')
plt.show()
```




```
churners =
last_purchase_customers[last_purchase_customers['DaysSinceLastPurchase
'] > 365]
print("Potential Churners:\n", churners)

Potential Churners:
     CustomerID     TransactionDate  DaysSinceLastPurchase
13       C0014 2024-01-17 18:31:55                    372
109      C0110 2024-01-02 19:11:34                    387
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

transactions = pd.read_csv('Transactions.csv')
products = pd.read_csv('Products.csv')
customers = pd.read_csv('Customers.csv')

#I am not checking for null types or missing values and such as i
checked for all those things in EDA part of Assignments
dataset = transactions.merge(products,on = "ProductID",how =
'left').merge(customers,on = "CustomerID",how = 'left')
dataset['TransactionDate'] =
pd.to_datetime(dataset['TransactionDate'])
dataset['SignupDate'] = pd.to_datetime(dataset['SignupDate'])
dataset = dataset.drop(columns='Price_y',axis=1)
dataset.rename(columns={'Price_x':'Price'},inplace=True)
print(dataset)
```

```
    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 07:38:55         1

3          T00272      C0087      P067 2024-03-26 22:55:37         2

4          T00363      C0070      P067 2024-03-21 15:10:10         3

..            ...        ...       ...                 ...       ...

995        T00496      C0118      P037 2024-10-24 08:30:27         1

996        T00759      C0059      P037 2024-06-04 02:15:24         3

997        T00922      C0018      P037 2024-04-05 13:05:32         4

998        T00959      C0115      P037 2024-09-29 10:16:02         2

999        T00992      C0024      P037 2024-04-21 10:52:24         1


     TotalValue   Price                      ProductName     Category
\
0        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics

1        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics

2        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics
```

```
3        601.36  300.68  ComfortLiving Bluetooth Speaker  Electronics
4        902.04  300.68  ComfortLiving Bluetooth Speaker  Electronics
..          ...     ...                              ...          ...
995      459.86  459.86              SoundWave Smartwatch  Electronics
996     1379.58  459.86              SoundWave Smartwatch  Electronics
997     1839.44  459.86              SoundWave Smartwatch  Electronics
998      919.72  459.86              SoundWave Smartwatch  Electronics
999      459.86  459.86              SoundWave Smartwatch  Electronics

             CustomerName         Region SignupDate
0          Andrea Jenkins         Europe 2022-12-03
1         Brittany Harvey           Asia 2024-09-04
2         Kathryn Stevens         Europe 2024-04-04
3         Travis Campbell  South America 2024-04-11
4           Timothy Perez         Europe 2022-03-15
..                    ...            ...        ...
995            Jacob Holt  South America 2022-01-22
996  Mrs. Kimberly Wright  North America 2024-04-07
997          Tyler Haynes  North America 2024-09-21
998       Joshua Hamilton           Asia 2024-11-11
999        Michele Cooley  North America 2024-02-05

[1000 rows x 12 columns]
```

```
dataset.describe()
```

{"summary":"{\n \"name\": \"dataset\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"TransactionDate\",\n   \"properties\": {\n   \"dtype\": \"date\",\n   \"min\": \"1970-01-01 00:00:00.000001\",\n   \"max\": \"2024-12-28 11:00:00\",\n   \"num_unique_values\": 7,\n   \"samples\": [\n   \"1000\",\n   \"2024-06-23 15:33:02.768999936\",\n   \"2024-09-19 14:19:57\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Quantity\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 352.66353426013046,\n   \"min\": 1.0,\n   \"max\": 1000.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   1000.0,\n   2.537,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"TotalValue\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 598.9454831884048,\n   \"min\": 16.08,\n   \"max\":

```
1991.04,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
689.9955600000001,\n          1011.66,\n          1000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 305.16091561989646,\
n        \"min\": 16.08,\n        \"max\": 1000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
272.55407,\n          404.4,\n          1000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SignupDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"1970-01-01 00:00:00.000001\",\n        \"max\": \"2024-12-28
00:00:00\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"1000\",\n          \"2023-07-09 02:49:55.199999744\",\n
\"2024-04-12 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
regions_encoded = pd.get_dummies(dataset[['CustomerID','Region']],
columns=['Region'])
print(regions_encoded.columns)
```

```
Index(['CustomerID', 'Region_Asia', 'Region_Europe', 'Region_North
America',
       'Region_South America'],
      dtype='object')
```

```
customer_aggregates = dataset.groupby('CustomerID').agg(
    TotalTransactions=('TransactionID', 'count'),
    TotalQuantity=('Quantity', 'sum'),
    TotalValue=('TotalValue', 'sum'),
    AvgSpendingPerTransaction = ('TotalValue', lambda x: x.sum() /
x.count())
).reset_index()
print(customer_aggregates)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                  5             12     3354.52
1        C0002                  4             10     1862.74
2        C0003                  4             14     2725.38
3        C0004                  8             23     5354.88
4        C0005                  3              7     2034.24
..         ...                ...            ...         ...
194      C0196                  4             12     4982.88
195      C0197                  3              9     1928.65
196      C0198                  2              3      931.83
197      C0199                  4              9     1979.28
198      C0200                  5             16     4758.60

     AvgSpendingPerTransaction
```

```
0                  670.904000
1                  465.685000
2                  681.345000
3                  669.360000
4                  678.080000
..                        ...
194               1245.720000
195                642.883333
196                465.915000
197                494.820000
198                951.720000

[199 rows x 5 columns]
```

```
category_pivot = dataset.pivot_table(
    index='CustomerID',
    columns='Category',
     values='Quantity',
    aggfunc='sum',
    fill_value=0
).reset_index()
print(category_pivot)
```

```
Category CustomerID  Books  Clothing  Electronics  Home Decor
0             C0001      2         0            7           3
1             C0002      0         4            0           6
2             C0003      0         4            4           6
3             C0004      8         0            6           9
4             C0005      0         0            4           3
..              ...    ...       ...          ...         ...
194           C0196      3         4            0           5
195           C0197      0         0            6           3
196           C0198      0         2            1           0
197           C0199      0         0            3           6
198           C0200      4         7            1           4

[199 rows x 5 columns]
```

```
customer_features =
customer_aggregates.merge(regions_encoded,on='CustomerID',how='left').
merge(category_pivot,on='CustomerID',how='left')
customer_features =
customer_features.drop_duplicates(subset='CustomerID').reset_index()
customer_features = customer_features.drop(columns='index',axis=1)
print(customer_features)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                  5             12     3354.52
1        C0002                  4             10     1862.74
2        C0003                  4             14     2725.38
```

```
3        C0004                  8              23     5354.88
4        C0005                  3               7     2034.24
..         ...                ...             ...         ...
194      C0196                  4              12     4982.88
195      C0197                  3               9     1928.65
196      C0198                  2               3      931.83
197      C0199                  4               9     1979.28
198      C0200                  5              16     4758.60

     AvgSpendingPerTransaction  Region_Asia  Region_Europe  \
0                   670.904000        False          False
1                   465.685000         True          False
2                   681.345000        False          False
3                   669.360000        False          False
4                   678.080000         True          False
..                         ...          ...            ...
194                1245.720000        False           True
195                 642.883333        False           True
196                 465.915000        False           True
197                 494.820000        False           True
198                 951.720000         True          False

     Region_North America  Region_South America  Books  Clothing
Electronics  \
0                   False                  True      2         0
7
1                   False                 False      0         4
0
2                   False                  True      0         4
4
3                   False                  True      8         0
6
4                   False                 False      0         0
4
..                    ...                   ...    ...       ...
...
194                 False                 False      3         4
0
195                 False                 False      0         0
6
196                 False                 False      0         2
1
197                 False                 False      0         0
3
198                 False                 False      4         7
1

     Home Decor
0             3
1             6
```

```
2              6
3              9
4              3
..           ...
194            5
195            3
196            0
197            6
198            4

[199 rows x 13 columns]
```

```
import datetime
last_purchase_customers = dataset.groupby('CustomerID')
['TransactionDate'].max().reset_index()
days_since_last_purchase = datetime.datetime.now() -
last_purchase_customers['TransactionDate']
customer_features['DaysSinceLastPurchase'] =
days_since_last_purchase.dt.days
print(customer_features['DaysSinceLastPurchase'])
```

```
0       82
1       52
2      152
3       32
4       81
      ...
194     40
195     27
196    111
197     90
198     44
Name: DaysSinceLastPurchase, Length: 199, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
customer_features['TotalValue'] =
scaler.fit_transform(customer_features[['TotalValue']])
customer_features['DaysSinceLastPurchase'] =
scaler.fit_transform(customer_features[['DaysSinceLastPurchase']])
customer_features['TotalTransactions'] =
scaler.fit_transform(customer_features[['TotalTransactions']])
customer_features['TotalQuantity'] =
scaler.fit_transform(customer_features[['TotalQuantity']])
customer_features['AvgSpendingPerTransaction'] =
scaler.fit_transform(customer_features[['AvgSpendingPerTransaction']])

print(customer_features)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                0.4       0.354839    0.308942   
1        C0002                0.3       0.290323    0.168095   
2        C0003                0.3       0.419355    0.249541   
3        C0004                0.7       0.709677    0.497806   
4        C0005                0.2       0.193548    0.184287   
..         ...                ...            ...         ...   
194      C0196                0.3       0.354839    0.462684   
195      C0197                0.2       0.258065    0.174318   
196      C0198                0.1       0.064516    0.080203   
197      C0199                0.3       0.258065    0.179098   
198      C0200                0.4       0.483871    0.441508   

     AvgSpendingPerTransaction  Region_Asia  Region_Europe  \
0                     0.474336        False          False   
1                     0.308940         True          False   
2                     0.482751        False          False   
3                     0.473092        False          False   
4                     0.480120         True          False   
..                         ...          ...            ...   
194                   0.937609        False           True   
195                   0.451753        False           True   
196                   0.309126        False           True   
197                   0.332422        False           True   
198                   0.700660         True          False   

     Region_North America  Region_South America  Books  Clothing
Electronics  \
0                   False                  True      2         0
7   
1                   False                 False      0         4
0   
2                   False                  True      0         4
4   
3                   False                  True      8         0
6   
4                   False                 False      0         0
4   
..                    ...                   ...    ...       ...
...   
194                 False                 False      3         4
0   
195                 False                 False      0         0
6   
196                 False                 False      0         2
1   
197                 False                 False      0         0
3   
198                 False                 False      4         7
1   
```

```
     Home Decor  DaysSinceLastPurchase
0             3               0.152778
1             6               0.069444
2             6               0.347222
3             9               0.013889
4             3               0.150000
..          ...                    ...
194           5               0.036111
195           3               0.000000
196           0               0.233333
197           6               0.175000
198           4               0.047222

[199 rows x 14 columns]
```

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_matrix =
cosine_similarity(customer_features.drop(columns='CustomerID'))
print(similarity_matrix)
```

```
[[1.         0.3164972  0.71527379 ... 0.36689163 0.72661002
0.38017404]
 [0.3164972  1.         0.86024079 ... 0.45070416 0.73066705
0.79996009]
 [0.71527379 0.86024079 1.         ... 0.59352677 0.85296441
0.74189584]
 ...
 [0.36689163 0.45070416 0.59352677 ... 1.         0.24867467
0.67320372]
 [0.72661002 0.73066705 0.85296441 ... 0.24867467 1.
0.44164937]
 [0.38017404 0.79996009 0.74189584 ... 0.67320372 0.44164937 1.
]]
```

```
customer_similarity_df = pd.DataFrame(
    similarity_matrix,
    index=customer_features['CustomerID'],
    columns=customer_features.index
)
print(customer_similarity_df)
```

```
                 0         1         2         3         4         5
\
CustomerID

C0001       1.000000  0.316497  0.715274  0.804402  0.914503  0.361022

C0002       0.316497  1.000000  0.860241  0.552602  0.514999  0.655393

C0003       0.715274  0.860241  1.000000  0.706350  0.804239  0.594429
```

```
C0004        0.804402  0.552602  0.706350  1.000000  0.741906  0.706496

C0005        0.914503  0.514999  0.804239  0.741906  1.000000  0.259506

...               ...       ...       ...       ...       ...       ...

C0196        0.378539  0.881803  0.775484  0.715931  0.421942  0.903547

C0197        0.947296  0.368284  0.746670  0.690015  0.953806  0.194975

C0198        0.366892  0.450704  0.593527  0.187395  0.331694  0.467662

C0199        0.726610  0.730667  0.852964  0.787917  0.867792  0.377846

C0200        0.380174  0.799960  0.741896  0.605266  0.373779  0.922571


                    6         7         8         9  ...       189
190  \
CustomerID                                           ...

C0001        0.876922  0.617688  0.366805  0.054279  ...  0.452373
0.848767
C0002        0.599253  0.829177  0.451584  0.538016  ...  0.005770
0.113010
C0003        0.839077  0.938409  0.593609  0.472173  ...  0.122877
0.492436
C0004        0.772664  0.608449  0.188202  0.119935  ...  0.681372
0.699108
C0005        0.989609  0.661823  0.329101  0.005854  ...  0.175882
0.608223
...               ...       ...       ...       ...  ...       ...
...
C0196        0.496869  0.804427  0.518935  0.640637  ...  0.412562
0.358774
C0197        0.922211  0.629121  0.424284  0.018676  ...  0.195828
0.681276
C0198        0.300324  0.755034  0.997687  0.830710  ...  0.096174
0.481237
C0199        0.922503  0.668922  0.248459  0.018926  ...  0.101071
0.346768
C0200        0.408816  0.866374  0.673136  0.834792  ...  0.454118
0.490551

                  191       192       193       194       195       196
\
CustomerID

C0001        0.256306  0.813235  0.576731  0.378539  0.947296  0.366892
```

```
C0002        0.017770  0.694193  0.944028  0.881803  0.368284  0.450704
C0003        0.008236  0.876698  0.971983  0.775484  0.746670  0.593527
C0004        0.591594  0.872362  0.690567  0.715931  0.690015  0.187395
C0005        0.026739  0.770271  0.711740  0.421942  0.953806  0.331694
...               ...       ...       ...       ...       ...       ...
C0196        0.422266  0.803761  0.843462  1.000000  0.338289  0.522550
C0197        0.005768  0.743899  0.618392  0.338289  1.000000  0.425206
C0198        0.013770  0.574558  0.494562  0.522550  0.425206  1.000000
C0199        0.006715  0.744990  0.863221  0.641537  0.805172  0.248675
C0200        0.450556  0.816010  0.756037  0.922111  0.297947  0.673204

                  197       198
CustomerID
C0001        0.726610  0.380174
C0002        0.730667  0.799960
C0003        0.852964  0.741896
C0004        0.787917  0.605266
C0005        0.867792  0.373779
...               ...       ...
C0196        0.641537  0.922111
C0197        0.805172  0.297947
C0198        0.248675  0.673204
C0199        1.000000  0.441649
C0200        0.441649  1.000000

[199 rows x 199 columns]
```

```
top_3_similar_customers = customer_similarity_df.apply(
    lambda row:
list(zip(row.sort_values(ascending=False).iloc[1:4].index.tolist(),
                         [round(score, 3) for score in
row.sort_values(ascending=False).iloc[1:4].tolist()])),
    axis=1
)
recommendations = pd.DataFrame({
    'CustomerID': top_3_similar_customers.index,
    'Top_3_Similar_Customers_with_Similarity_Score':
top_3_similar_customers
})
recommendations
recommendations.head(n=20)
```

{"repr_error":"cannot insert CustomerID, already exists","type":"dataframe","variable_name":"recommendations"}

```
recommendations.head(n=20).to_csv('Lookalike.csv',index=False)
```

```
print(customer_features.columns)
```

```
Index(['CustomerID', 'TotalTransactions', 'TotalQuantity',
'TotalValue',
       'AvgSpendingPerTransaction', 'Region_Asia', 'Region_Europe',
       'Region_North America', 'Region_South America', 'Books',
'Clothing',
       'Electronics', 'Home Decor', 'DaysSinceLastPurchase'],
      dtype='object')
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

transactions = pd.read_csv('Transactions.csv')
products = pd.read_csv('Products.csv')
customers = pd.read_csv('Customers.csv')

#I am not checking for null types or missing values and such as i
checked for all those things in EDA part of Assignments
dataset = transactions.merge(products,on = "ProductID",how =
'left').merge(customers,on = "CustomerID",how = 'left')
dataset['TransactionDate'] =
pd.to_datetime(dataset['TransactionDate'])
dataset['SignupDate'] = pd.to_datetime(dataset['SignupDate'])
dataset = dataset.drop(columns='Price_y',axis=1)
dataset.rename(columns={'Price_x':'Price'},inplace=True)
print(dataset)
```

```
    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 07:38:55         1

3          T00272      C0087      P067 2024-03-26 22:55:37         2

4          T00363      C0070      P067 2024-03-21 15:10:10         3

..            ...        ...       ...                 ...       ...

995        T00496      C0118      P037 2024-10-24 08:30:27         1

996        T00759      C0059      P037 2024-06-04 02:15:24         3

997        T00922      C0018      P037 2024-04-05 13:05:32         4

998        T00959      C0115      P037 2024-09-29 10:16:02         2

999        T00992      C0024      P037 2024-04-21 10:52:24         1


     TotalValue   Price                      ProductName     Category
\
0        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics

1        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics

2        300.68  300.68  ComfortLiving Bluetooth Speaker  Electronics
```

```
3        601.36  300.68  ComfortLiving Bluetooth Speaker  Electronics
4        902.04  300.68  ComfortLiving Bluetooth Speaker  Electronics
..          ...     ...                              ...          ...
995      459.86  459.86              SoundWave Smartwatch  Electronics
996     1379.58  459.86              SoundWave Smartwatch  Electronics
997     1839.44  459.86              SoundWave Smartwatch  Electronics
998      919.72  459.86              SoundWave Smartwatch  Electronics
999      459.86  459.86              SoundWave Smartwatch  Electronics

             CustomerName         Region SignupDate
0          Andrea Jenkins         Europe 2022-12-03
1         Brittany Harvey           Asia 2024-09-04
2         Kathryn Stevens         Europe 2024-04-04
3         Travis Campbell  South America 2024-04-11
4           Timothy Perez         Europe 2022-03-15
..                    ...            ...        ...
995            Jacob Holt  South America 2022-01-22
996  Mrs. Kimberly Wright  North America 2024-04-07
997          Tyler Haynes  North America 2024-09-21
998       Joshua Hamilton           Asia 2024-11-11
999        Michele Cooley  North America 2024-02-05

[1000 rows x 12 columns]
```

```
dataset.describe()
```

{"summary":"{\n \"name\": \"dataset\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"TransactionDate\",\n   \"properties\": {\n   \"dtype\": \"date\",\n   \"min\": \"1970-01-01 00:00:00.000001\",\n   \"max\": \"2024-12-28 11:00:00\",\n   \"num_unique_values\": 7,\n   \"samples\": [\n   \"1000\",\n   \"2024-06-23 15:33:02.768999936\",\n   \"2024-09-19 14:19:57\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Quantity\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 352.66353426013046,\n   \"min\": 1.0,\n   \"max\": 1000.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   1000.0,\n   2.537,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"TotalValue\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 598.9454831884048,\n   \"min\": 16.08,\n   \"max\":

```
1991.04,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
689.9955600000001,\n          1011.66,\n          1000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 305.16091561989646,\
n        \"min\": 16.08,\n        \"max\": 1000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
272.55407,\n          404.4,\n          1000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SignupDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"1970-01-01 00:00:00.000001\",\n        \"max\": \"2024-12-28
00:00:00\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"1000\",\n          \"2023-07-09 02:49:55.199999744\",\n
\"2024-04-12 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```



```
regions_encoded = pd.get_dummies(dataset[['CustomerID','Region']],
columns=['Region'])
print(regions_encoded.columns)
```

```
Index(['CustomerID', 'Region_Asia', 'Region_Europe', 'Region_North
America',
       'Region_South America'],
      dtype='object')
```

```
customer_aggregates = dataset.groupby('CustomerID').agg(
    TotalTransactions=('TransactionID', 'count'),
    TotalQuantity=('Quantity', 'sum'),
    TotalValue=('TotalValue', 'sum'),
    AvgSpendingPerTransaction = ('TotalValue', lambda x: x.sum() /
x.count())
).reset_index()
print(customer_aggregates)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                  5             12     3354.52
1        C0002                  4             10     1862.74
2        C0003                  4             14     2725.38
3        C0004                  8             23     5354.88
4        C0005                  3              7     2034.24
..         ...                ...            ...         ...
194      C0196                  4             12     4982.88
195      C0197                  3              9     1928.65
196      C0198                  2              3      931.83
197      C0199                  4              9     1979.28
198      C0200                  5             16     4758.60

     AvgSpendingPerTransaction
```

```
0                 670.904000
1                 465.685000
2                 681.345000
3                 669.360000
4                 678.080000
..                       ...
194              1245.720000
195               642.883333
196               465.915000
197               494.820000
198               951.720000

[199 rows x 5 columns]
```

```
customer_features =
customer_aggregates.merge(regions_encoded,on='CustomerID',how='left')
customer_features =
customer_features.drop_duplicates(subset='CustomerID').reset_index()
customer_features = customer_features.drop(columns='index',axis=1)
print(customer_features)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                  5             12     3354.52
1        C0002                  4             10     1862.74
2        C0003                  4             14     2725.38
3        C0004                  8             23     5354.88
4        C0005                  3              7     2034.24
..         ...                ...            ...         ...
194      C0196                  4             12     4982.88
195      C0197                  3              9     1928.65
196      C0198                  2              3      931.83
197      C0199                  4              9     1979.28
198      C0200                  5             16     4758.60

     AvgSpendingPerTransaction  Region_Asia  Region_Europe  \
0                   670.904000        False          False
1                   465.685000         True          False
2                   681.345000        False          False
3                   669.360000        False          False
4                   678.080000         True          False
..                         ...          ...            ...
194                1245.720000        False           True
195                 642.883333        False           True
196                 465.915000        False           True
197                 494.820000        False           True
198                 951.720000         True          False

     Region_North America  Region_South America
0                   False                  True
1                   False                 False
```

```
2                    False                   True
3                    False                   True
4                    False                  False
..                     ...                    ...
194                  False                  False
195                  False                  False
196                  False                  False
197                  False                  False
198                  False                  False

[199 rows x 9 columns]
```

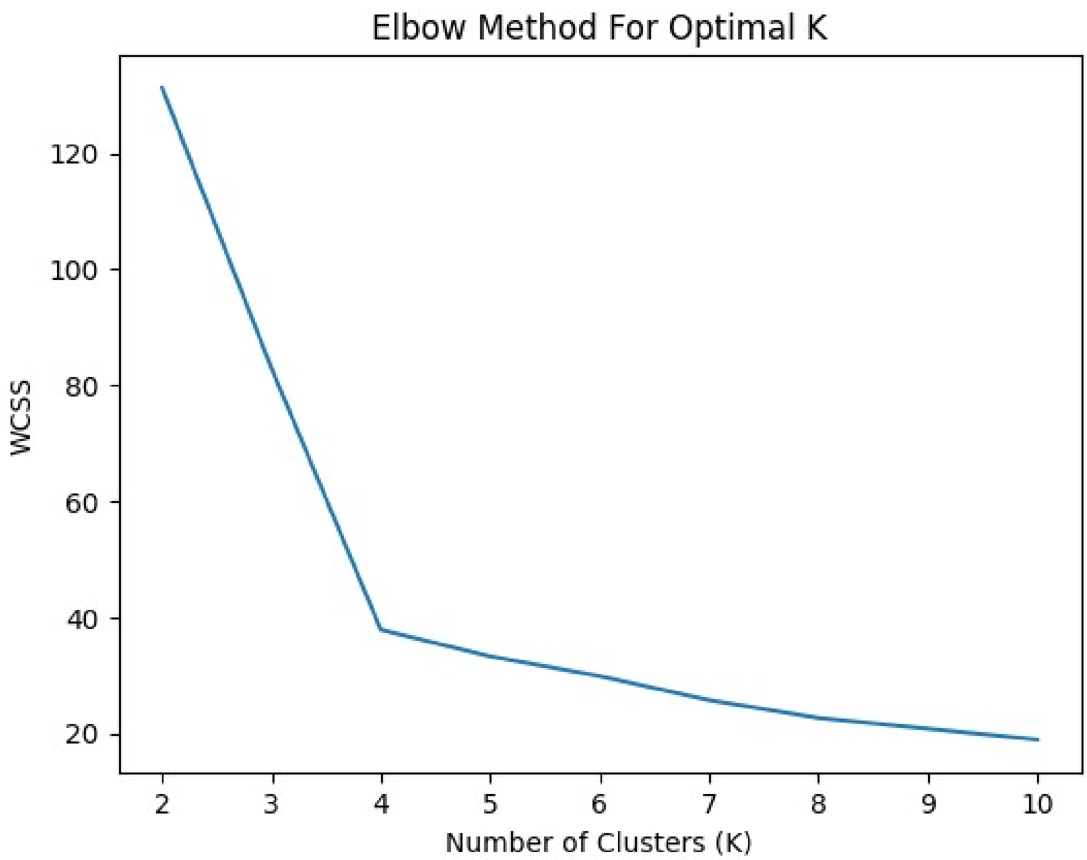
```
import datetime
last_purchase_customers = dataset.groupby('CustomerID')
['TransactionDate'].max().reset_index()
days_since_last_purchase = datetime.datetime.now() -
last_purchase_customers['TransactionDate']
customer_features['DaysSinceLastPurchase'] =
days_since_last_purchase.dt.days
print(customer_features['DaysSinceLastPurchase'])
```

```
0       82
1       52
2      152
3       32
4       81
      ...
194     40
195     27
196    111
197     90
198     44
Name: DaysSinceLastPurchase, Length: 199, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
customer_features['TotalValue'] =
scaler.fit_transform(customer_features[['TotalValue']])
customer_features['DaysSinceLastPurchase'] =
scaler.fit_transform(customer_features[['DaysSinceLastPurchase']])
customer_features['TotalTransactions'] =
scaler.fit_transform(customer_features[['TotalTransactions']])
customer_features['TotalQuantity'] =
scaler.fit_transform(customer_features[['TotalQuantity']])
customer_features['AvgSpendingPerTransaction'] =
scaler.fit_transform(customer_features[['AvgSpendingPerTransaction']])

print(customer_features)
```

```
    CustomerID  TotalTransactions  TotalQuantity  TotalValue  \
0        C0001                0.4       0.354839    0.308942   
1        C0002                0.3       0.290323    0.168095   
2        C0003                0.3       0.419355    0.249541   
3        C0004                0.7       0.709677    0.497806   
4        C0005                0.2       0.193548    0.184287   
..         ...                ...            ...         ...   
194      C0196                0.3       0.354839    0.462684   
195      C0197                0.2       0.258065    0.174318   
196      C0198                0.1       0.064516    0.080203   
197      C0199                0.3       0.258065    0.179098   
198      C0200                0.4       0.483871    0.441508   

     AvgSpendingPerTransaction  Region_Asia  Region_Europe  \
0                     0.474336        False          False   
1                     0.308940         True          False   
2                     0.482751        False          False   
3                     0.473092        False          False   
4                     0.480120         True          False   
..                         ...          ...            ...   
194                   0.937609        False           True   
195                   0.451753        False           True   
196                   0.309126        False           True   
197                   0.332422        False           True   
198                   0.700660         True          False   

     Region_North America  Region_South America  DaysSinceLastPurchase  

0                   False                  True               0.152778  

1                   False                 False               0.069444  

2                   False                  True               0.347222  

3                   False                  True               0.013889  

4                   False                 False               0.150000  

..                    ...                   ...                    ...  

194                 False                 False               0.036111  

195                 False                 False               0.000000  

196                 False                 False               0.233333  

197                 False                 False               0.175000  

198                 False                 False               0.047222  
```

```
[199 rows x 10 columns]

from sklearn.cluster import KMeans
wcss = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300,
n_init=10, random_state=42)
    kmeans.fit(customer_features.drop(columns=['CustomerID']))
    wcss.append(kmeans.inertia_)

plt.plot(range(2, 11), wcss)
plt.title('Elbow Method For Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.show()
```




```
from sklearn.metrics import silhouette_score,davies_bouldin_score
from sklearn.manifold import TSNE

param_grid = {
    'init': ['k-means++', 'random'],
    'max_iter': [80,90,100,110,120],
```

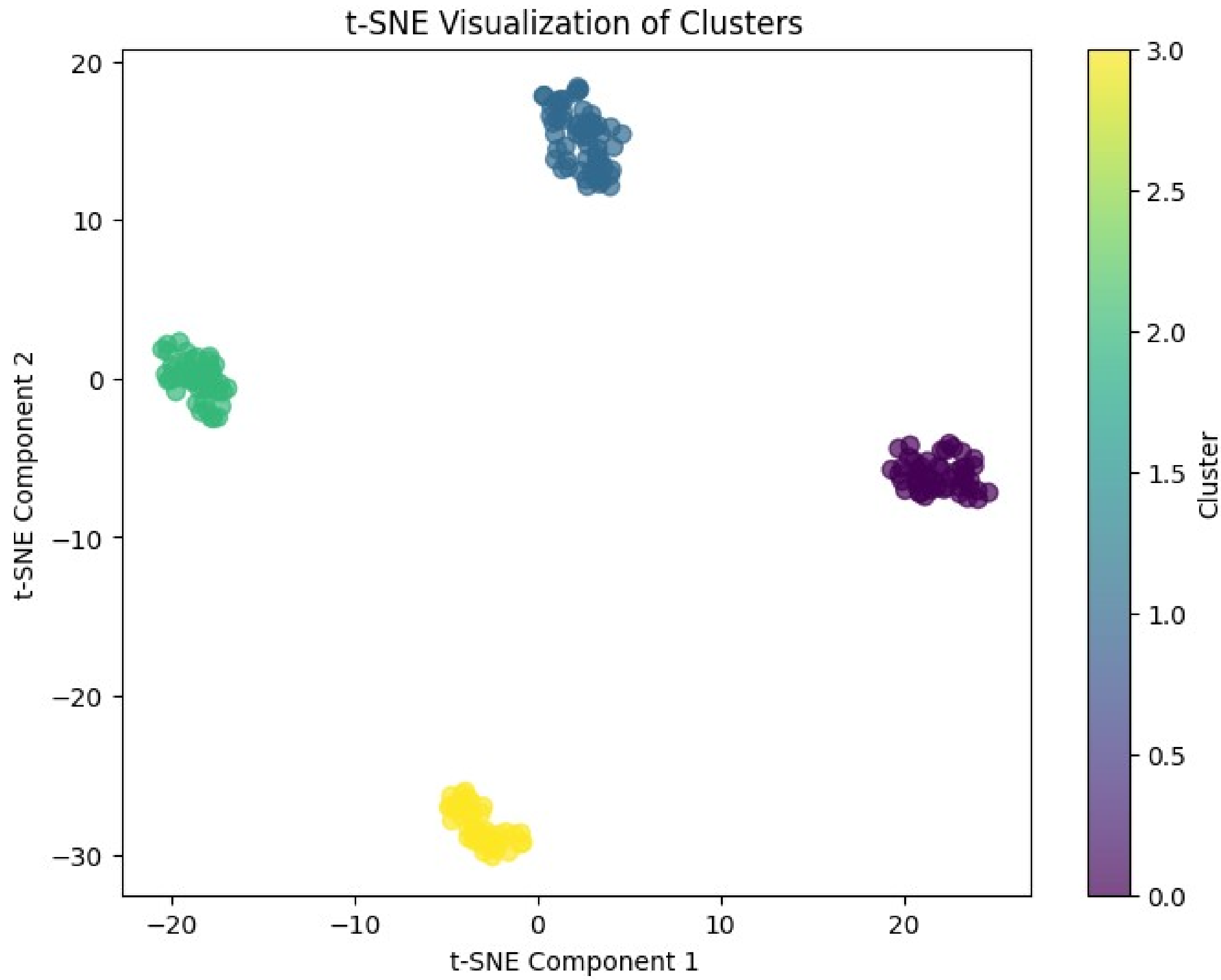
```
    'n_init': [3,4,5,6,7,8]
}

best_score = -1
best_params = {}
best_kmeans = None

X = customer_features.drop(columns=['CustomerID'])

for init in param_grid['init']:
    for max_iter in param_grid['max_iter']:
        for n_init in param_grid['n_init']:


            kmeans = KMeans(n_clusters=4, init=init,
max_iter=max_iter, n_init=n_init, random_state=42)
            kmeans.fit(X)

            silhouette_avg = silhouette_score(X, kmeans.labels_)
            print(f" init={init}, max_iter={max_iter},
n_init={n_init}, Silhouette Score={silhouette_avg}")
            if silhouette_avg > best_score:
                best_score = silhouette_avg
                best_params = {'init': init, 'max_iter': max_iter,
'n_init': n_init}
                best_kmeans = kmeans

print(f"Best Hyperparameters: {best_params}")
print(f"Best Silhouette Score: {best_score}")
db_index = davies_bouldin_score(X, kmeans.labels_)
print(f"Davies-Bouldin Index: {db_index}")

customer_features['Cluster'] = best_kmeans.labels_

tsne = TSNE(n_components=2, perplexity=30, random_state=42)
tsne_components = tsne.fit_transform(X)

tsne_df = pd.DataFrame(tsne_components, columns=['TSNE1', 'TSNE2'])
tsne_df['Cluster'] = customer_features['Cluster']

plt.figure(figsize=(8, 6))
plt.scatter(tsne_df['TSNE1'], tsne_df['TSNE2'], c=tsne_df['Cluster'],
cmap='viridis', s=50, alpha=0.7)
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.colorbar(label='Cluster')
plt.title('t-SNE Visualization of Clusters')
plt.show()
```

```
 init=k-means++, max_iter=80, n_init=3, Silhouette
Score=0.6983452304994472
```

```
 init=k-means++, max_iter=80, n_init=4, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=80, n_init=5, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=80, n_init=6, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=80, n_init=7, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=80, n_init=8, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=3, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=4, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=5, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=6, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=7, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=90, n_init=8, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=3, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=4, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=5, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=6, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=7, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=100, n_init=8, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=3, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=4, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=5, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=6, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=7, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=110, n_init=8, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=3, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=4, Silhouette
```

```
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=5, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=6, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=7, Silhouette
Score=0.6983452304994472
 init=k-means++, max_iter=120, n_init=8, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=3, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=4, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=5, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=6, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=7, Silhouette
Score=0.6983452304994472
 init=random, max_iter=80, n_init=8, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=3, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=4, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=5, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=6, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=7, Silhouette
Score=0.6983452304994472
 init=random, max_iter=90, n_init=8, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=3, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=4, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=5, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=6, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=7, Silhouette
Score=0.6983452304994472
 init=random, max_iter=100, n_init=8, Silhouette
Score=0.6983452304994472
 init=random, max_iter=110, n_init=3, Silhouette
Score=0.6983452304994472
 init=random, max_iter=110, n_init=4, Silhouette
Score=0.6983452304994472
```

```
 init=random, max_iter=110, n_init=5, Silhouette
Score=0.6983452304994472
 init=random, max_iter=110, n_init=6, Silhouette
Score=0.6983452304994472
 init=random, max_iter=110, n_init=7, Silhouette
Score=0.6983452304994472
 init=random, max_iter=110, n_init=8, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=3, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=4, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=5, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=6, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=7, Silhouette
Score=0.6983452304994472
 init=random, max_iter=120, n_init=8, Silhouette
Score=0.6983452304994472
Best Hyperparameters: {'init': 'k-means++', 'max_iter': 80, 'n_init':
3}
Best Silhouette Score: 0.6983452304994472
Davies-Bouldin Index: 0.449209015620059
```




```python
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=3, perplexity=30, random_state=42)
tsne_components = tsne.fit_transform(X)

tsne_df = pd.DataFrame(tsne_components, columns=['TSNE1', 'TSNE2',
'TSNE3'])
tsne_df['Cluster'] = customer_features['Cluster']


import plotly.express as px

fig = px.scatter_3d(
    tsne_df,
    x='TSNE1',
    y='TSNE2',
    z='TSNE3',
    color='Cluster',
    color_continuous_scale='Viridis',
```

```python
    title='3D t-SNE Visualization of Clusters'
)

fig.update_traces(marker=dict(size=5, opacity=0.8))
fig.show()
fig.write_html("3d_tsne_plot.html")
```

Business Insights Report

1. Executive Summary

- This report analyzes sales, customer behavior, and product performance data to uncover key insights that will inform strategic decisions. It highlights regional sales trends, top products, and customer acquisition and retention patterns, aiming to identify growth opportunities and optimize business strategies.
- Key highlights of findings:
  - South America has the largest customer base and also is the most revenue generating region.
  - Biggest Product Categories are Books followed by Electronics,Home Decor and Clothing in order.
  - Most Selling product is ActiveWear SmartWatch as well most revenue generating.

---

2. Objective

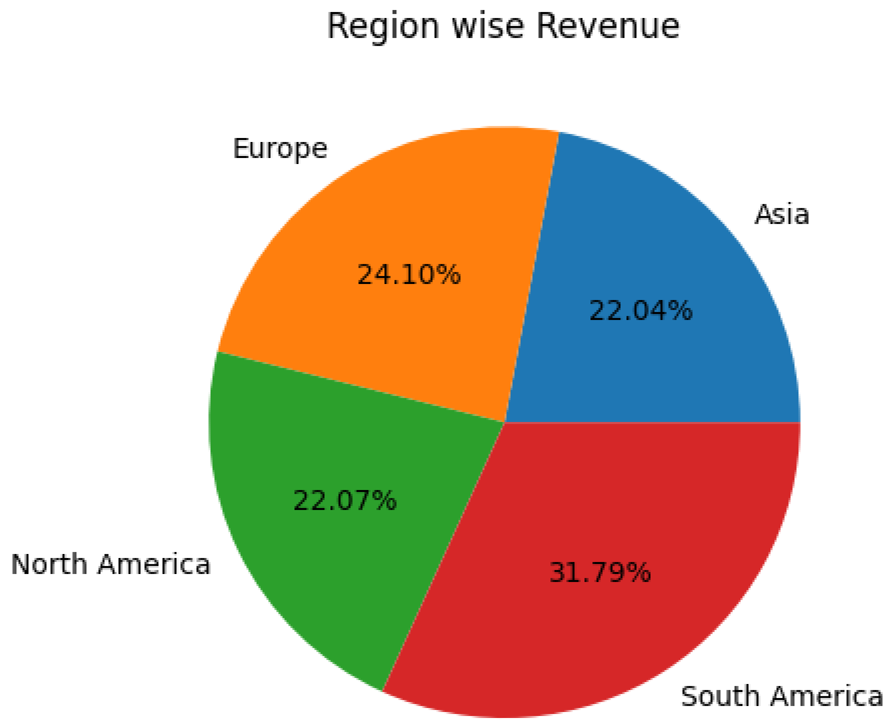
- To identify sales trends, customer behavior patterns, and revenue drivers to inform business strategy and decision-making.

---

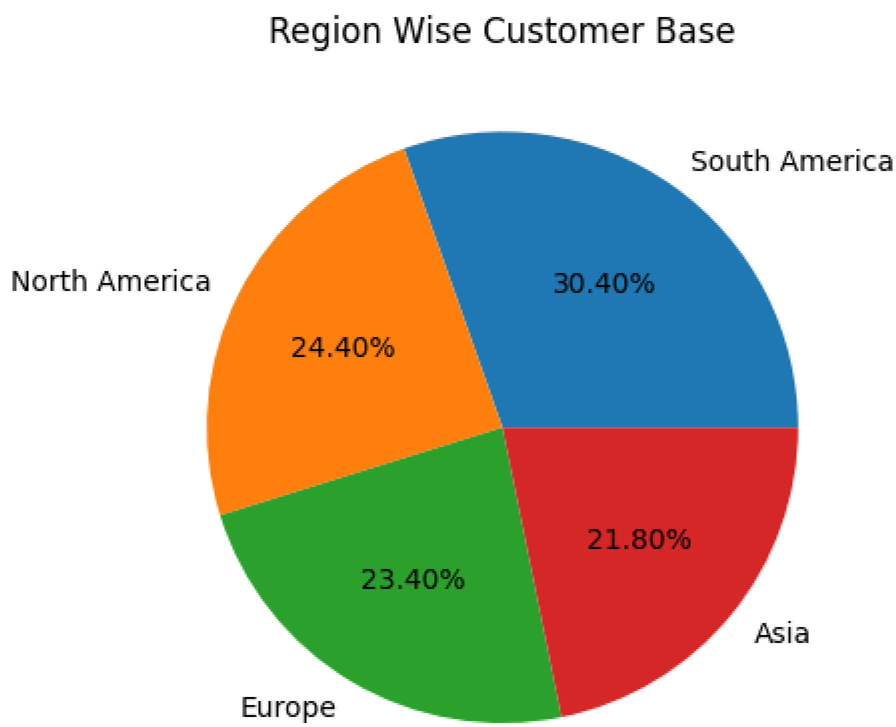
3. Key Business Insights

3.1 Revenue Analysis

- **Regional Performance**:
  - Top revenue-contributing region: South America.
  - Followed by Europe,North America and Asia in order.

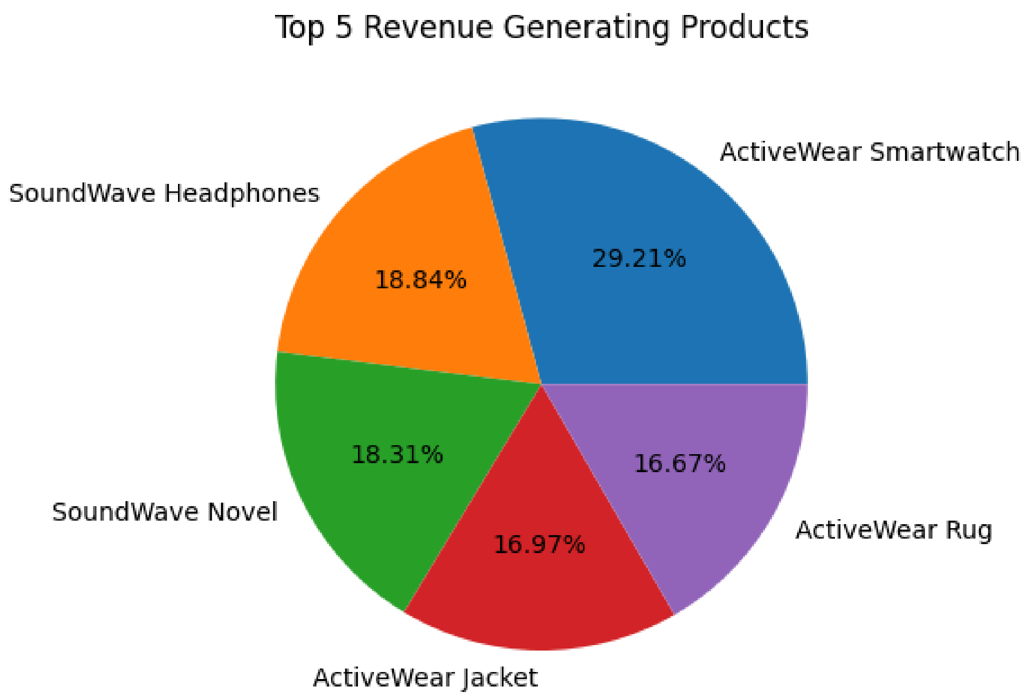


-
- South America has largest Customer Base followed by North America,Europe and Asia

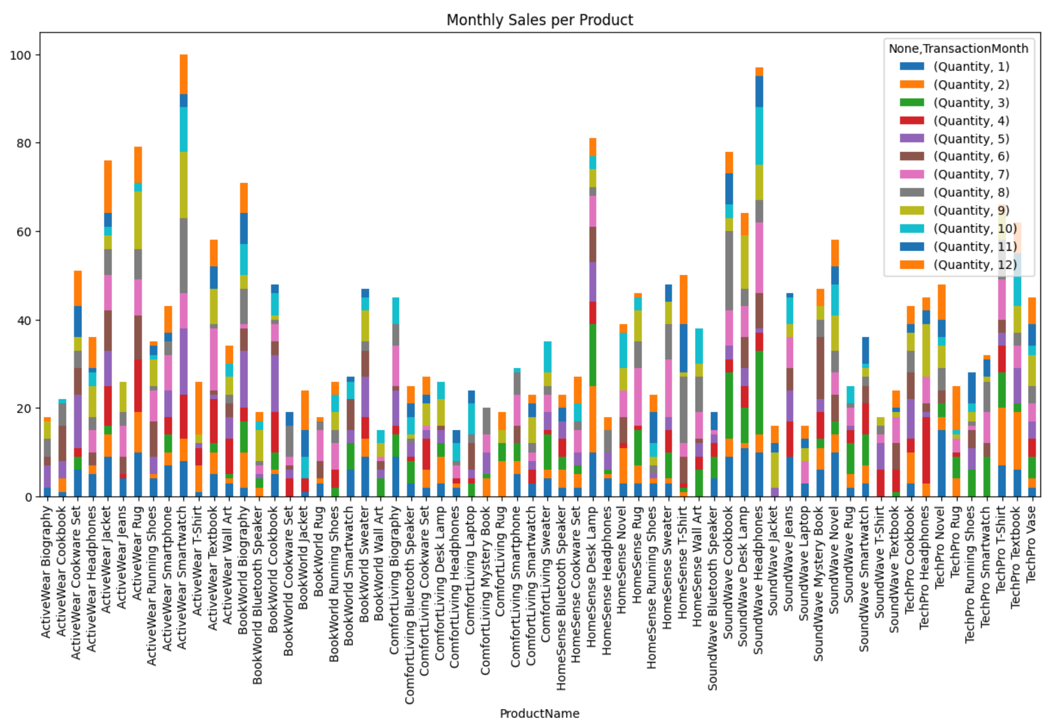


- ○
- ○ Insights: Asia market is untapped, an expansion in Asia could be considered.

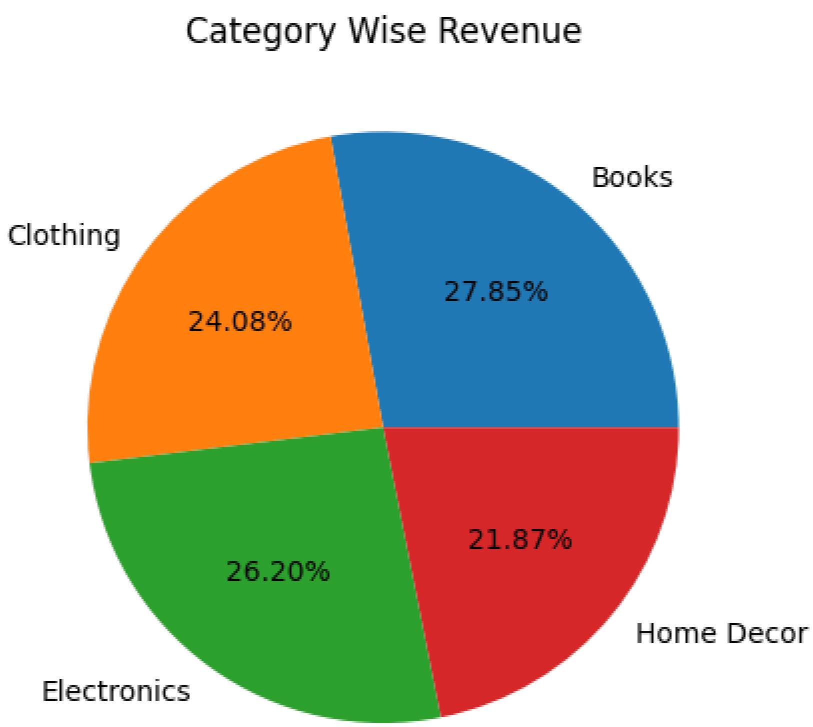
- **Product Analysis**:
  - Top revenue-generating product: Active Wear SmartWatch:



-
- Product Wise Monthly Sales:



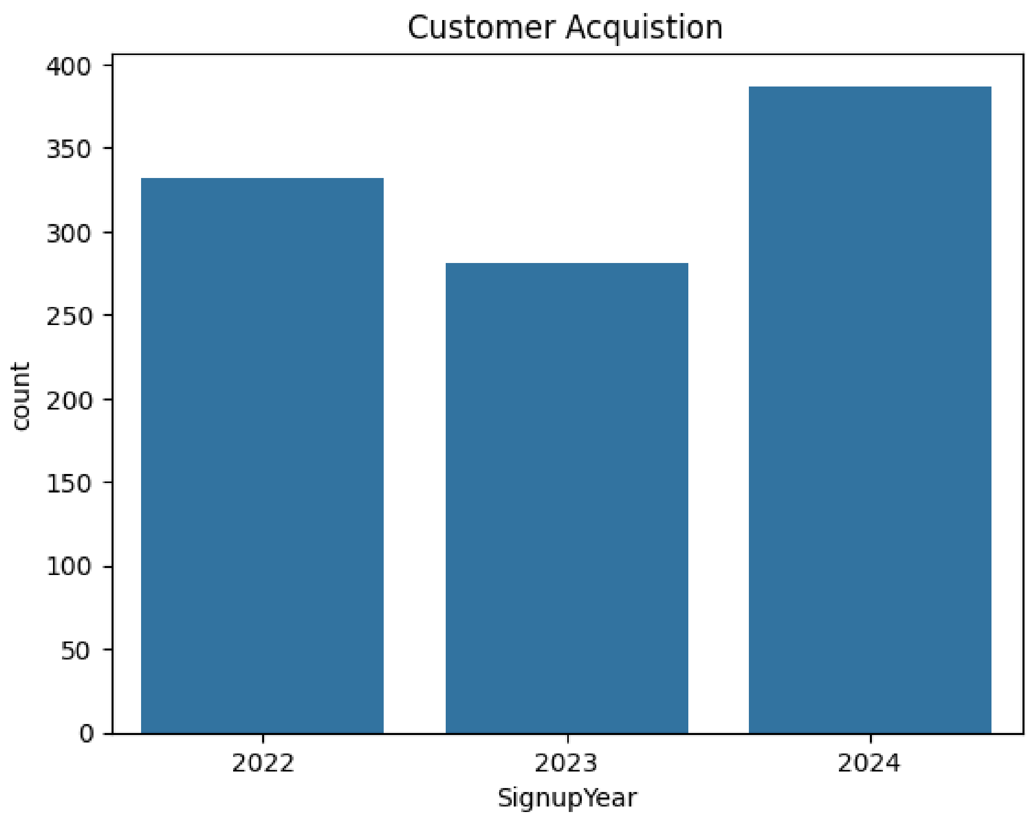
-
- Category-wise performance:



- Percentage contribution of the top 5 products:19.39%
- Insights:
  - ActiveWearBrand is trending; consider increasing its stock to meet demand.
  - Expand the inventory of books and run targeted promotions to capitalize on their strong performance.

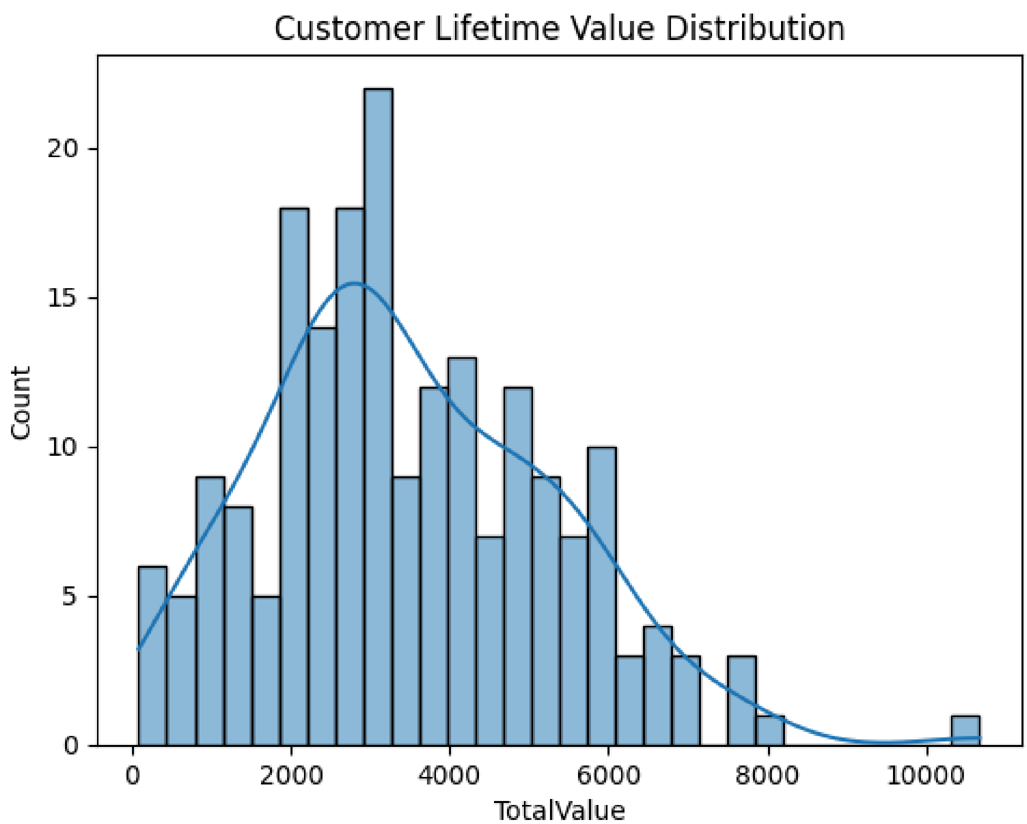
3.2 Customer Behavior Analysis

- **Customer Acquisition Trends**:
  - Growth in new customers over the years.
  - Customer Acquisition has gone up by 37.7% in the year 2024, but the year 2023 saw a decline of 15%.



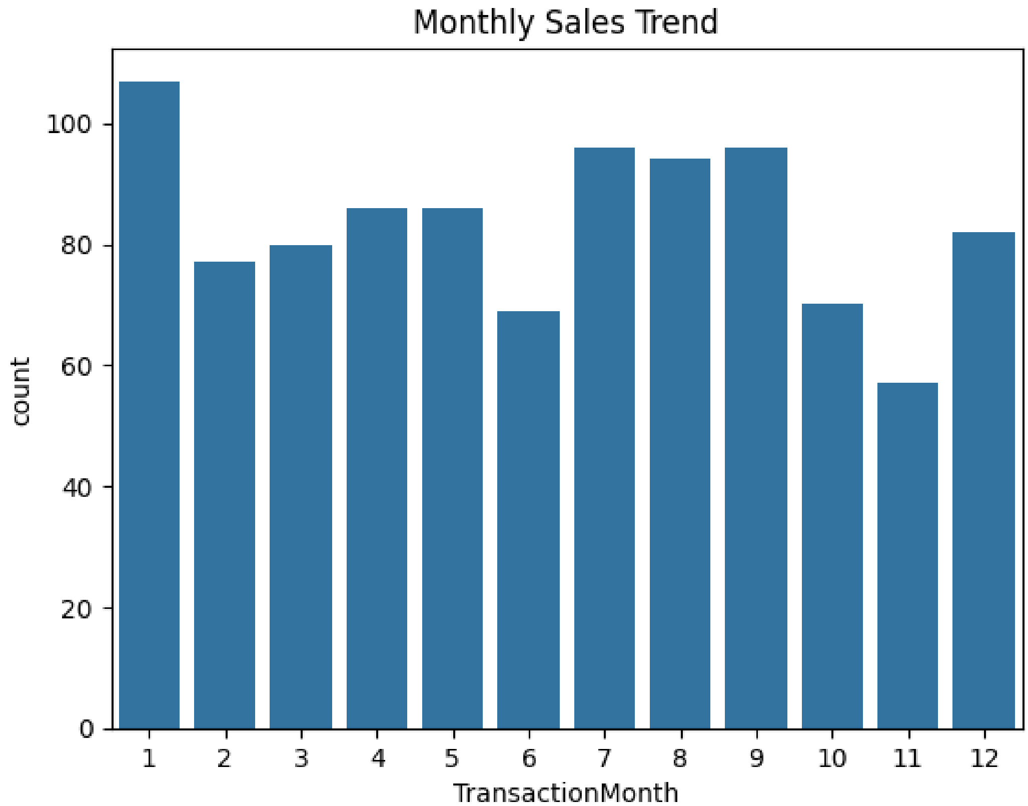
-
- Insights:
  - Focus on strategies that sustained the 37.7% growth in 2024 to counteract the previous year's 15% decline.

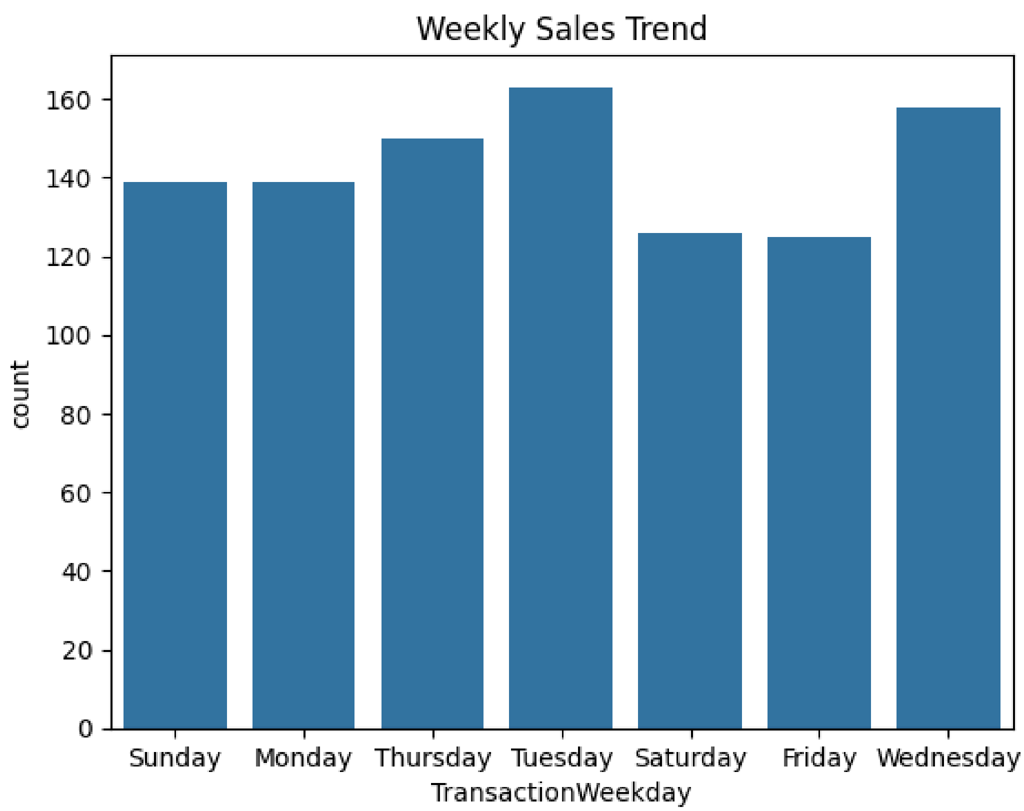
- **Customer Value Distribution:**



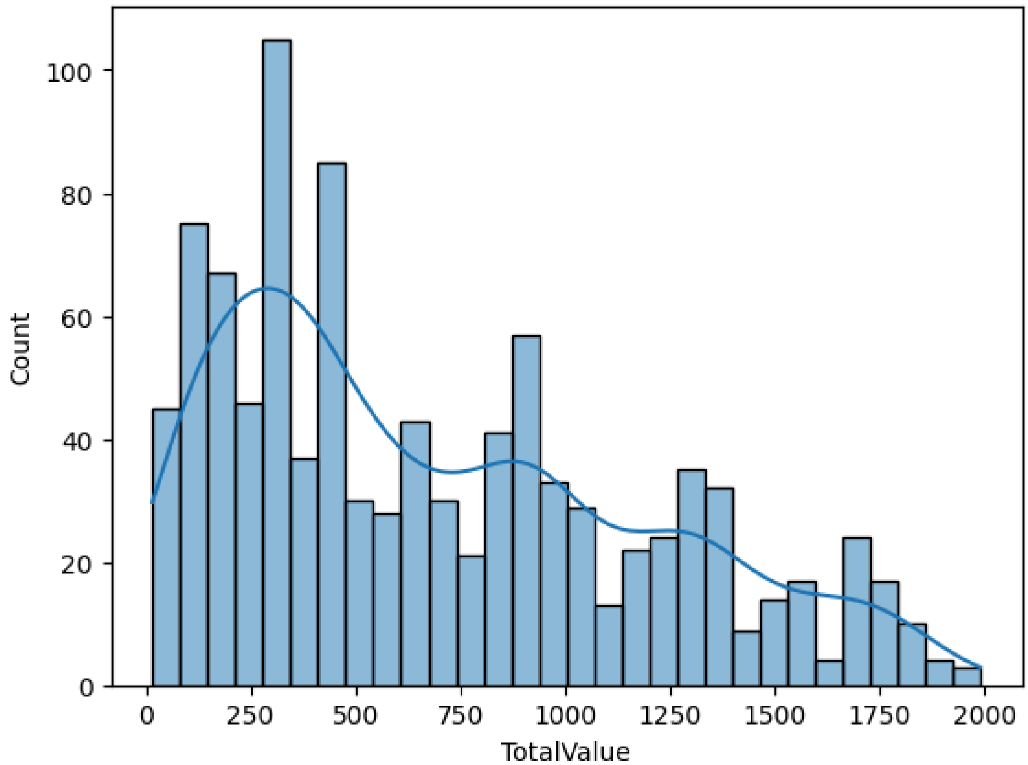
- The avg Repeat Purchase Rate: **78.89%**

- **Purchase Patterns**:
  - Monthly and weekday sales trends:



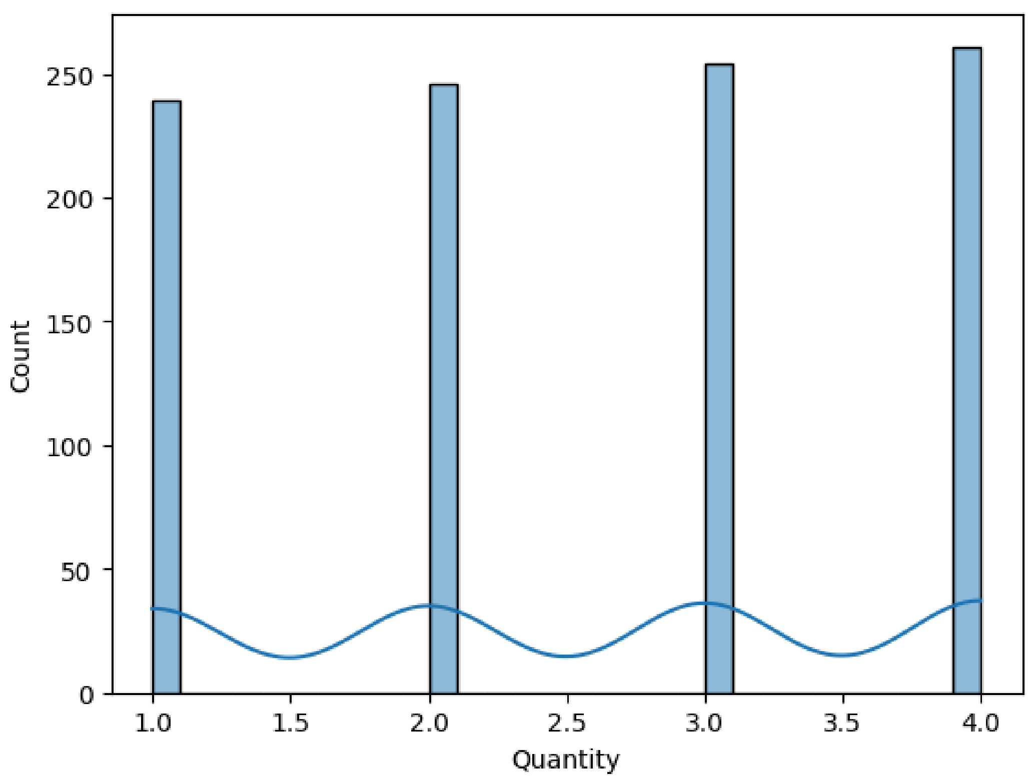


- Majority of Orders are in the 0-500$ range:



Count
TotalValue
0
20
40
60
80
100
0
250
500
750
1000
1250
1500
1750
2000

-



Count
Quantity
0
50
100
150
200
250
1.0
1.5
2.0
2.5
3.0
3.5
4.0

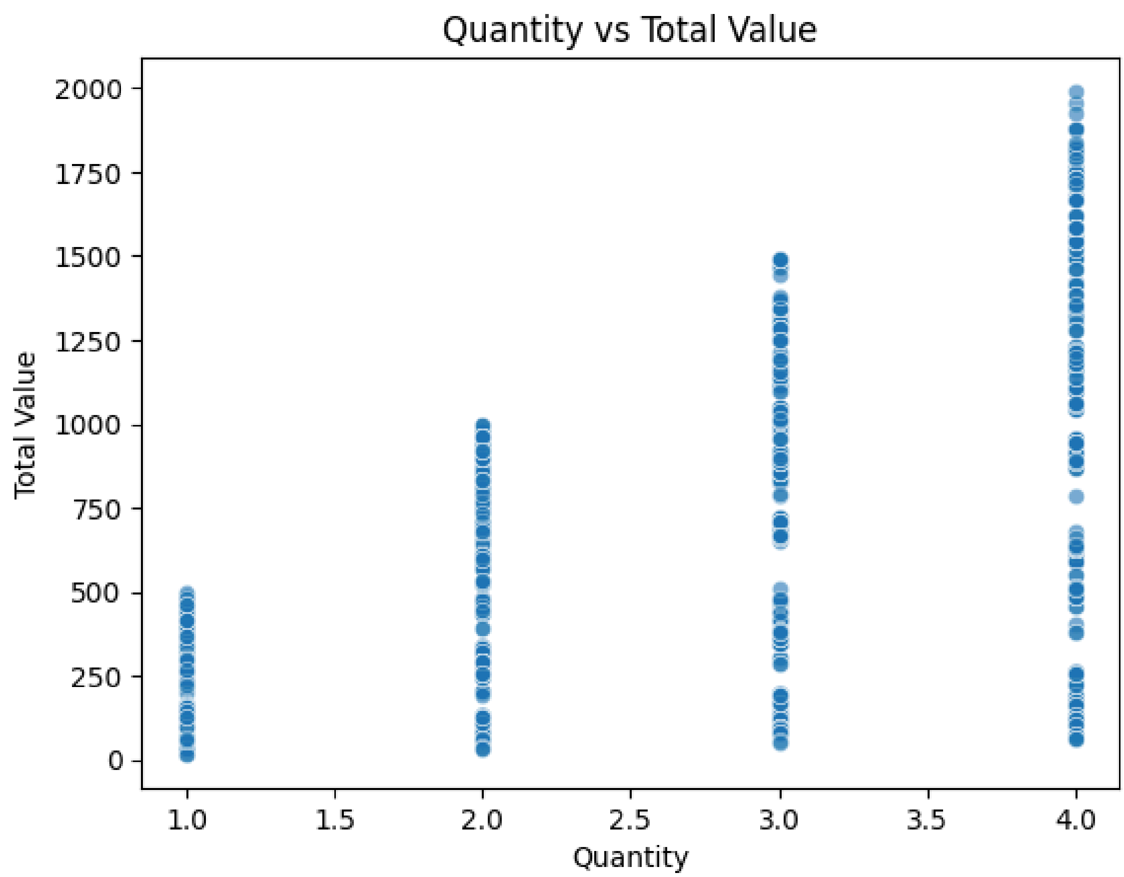
-
- Insight:

        - Highest Sales are seen in January probably due to New Year. July to September is also a great period for sales while November is the worst month for sales in the Year.
        - Weekends see relatively less transactions, whereas Weekdays are quite good for Sales.
- **Retention Trends**:

    - Customers who haven't purchased in over a year:
        - `Potential Churners:`
        - `     CustomerID     TransactionDate DaysSinceLastPurchase`
        - `13        C0014 2024-01-17 18:31:55                    372`
        - `109       C0110 2024-01-02 19:11:34                    387`
    - Suggestion: Target dormant customers with discount coupons or promotional offers.

3.3 Pricing and Quantity Insights

- Correlation between quantity purchased and total value:



-

- Insights:
    - Lower prices for bulk purchases could drive higher revenue.
    - Encourage bulk purchase promotions to maximize sales per customer.

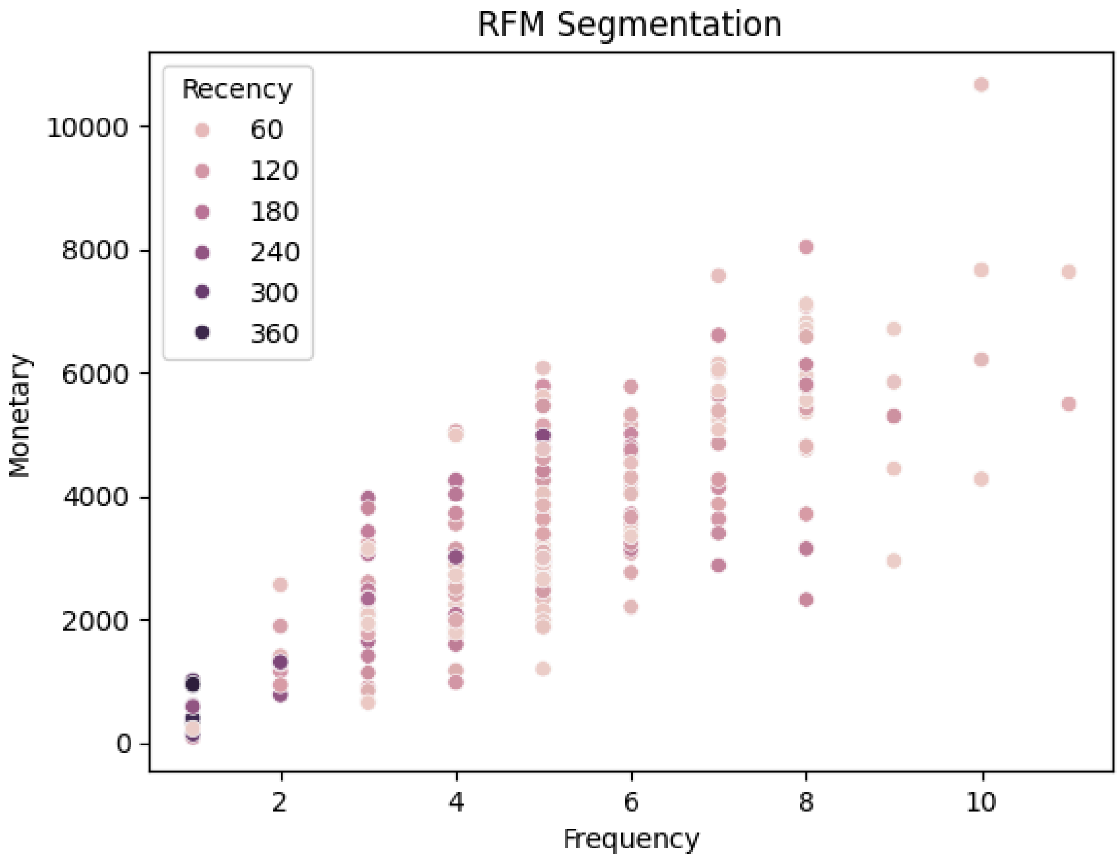
---

4. Recommendations

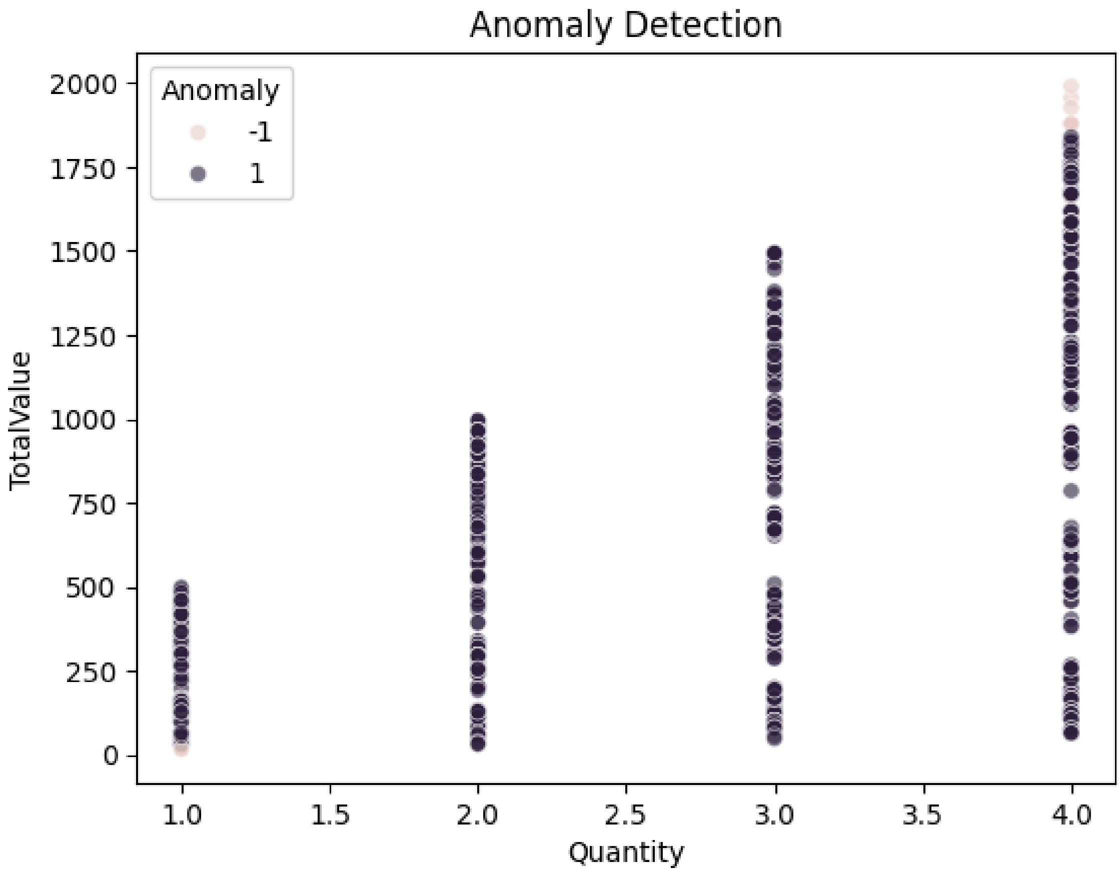
- **Targeted Marketing:**
    - Focus marketing efforts in South America, the highest revenue-generating region.
    - Re-engage dormant customers (inactive over a year) with personalized offers (discounts, loyalty points).
- **Product Optimization:**
    - Increase stock of top-selling products like ActiveWear Smartwatch.
    - Expand and promote Home Décor and Books to drive further growth.
- **Seasonal Campaigns:**
    - Launch aggressive marketing campaigns in January and July-September.
    - Run targeted promotions (e.g., Black Friday) to improve November sales.
- **Customer Retention:**
    - Use loyalty programs to retain frequent weekday shoppers.
    - Introduce weekend incentives (e.g., exclusive weekend deals) to boost sales.
- **Pricing Strategies:**
    - Offer bulk purchase discounts to drive higher sales per customer.
    - Run category-based promotions to maintain sales momentum year-round.
- **Behavioral Segmentation:**
    - Segment customers by purchase patterns and tailor offers based on their behavior (e.g., high spenders get VIP discounts).

---

5. Other Visual Insights


RFM Segmentation
Recency
60
120
180
240
300
360
10000
8000
6000
4000
2000
0
Monetary
2
4
6
8
10
Frequency

•



---

6. Limitations and Future Scope

- Constraints in the data or analysis:
    - Analysis does not include customer feedback or demographic details.

---

7. Conclusion

- This analysis highlights significant trends and opportunities for growth. South America remains the strongest market, with potential expansion opportunities in Asia. Books and ActiveWear Smartwatches drive the majority of sales, while customer acquisition has shown positive growth in 2024. Seasonal and bulk purchase patterns also present opportunities to optimize pricing and promotional strategies. By addressing these insights, the business can strengthen its market presence, enhance customer loyalty, and improve overall profitability.

-Mohak Kapoor

+91 9870232056
contact.mohakapoor@gmail.com
LinkedIn
GitHub

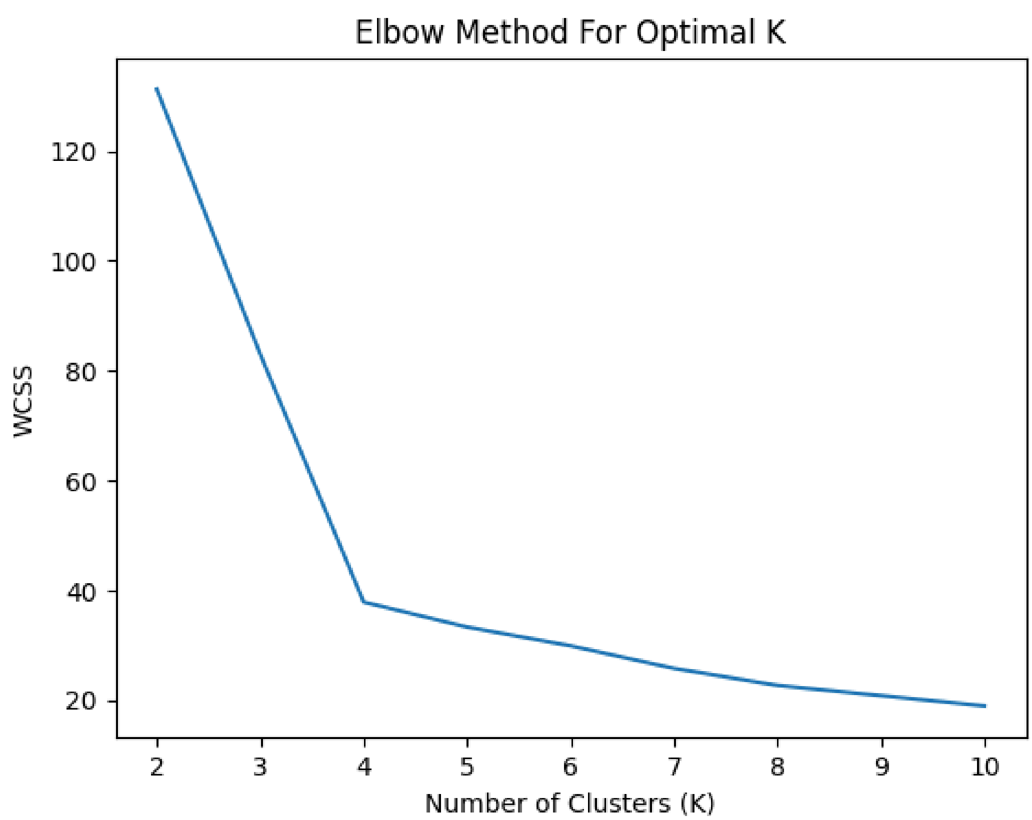
Clustering Analysis Report

1. Executive Summary

This report presents the findings from the clustering analysis performed on your dataset. The aim of the analysis was to segment customers to identify meaningful patterns and derive actionable insights. The clustering approach employed aims to group similar customers based on features such as customer data and transaction characteristics.

2. Clustering Methodology

- **Algorithm Used:** k-means++
- **Features Considered:** Based on your dataset, features such as **Transaction Amount**, **Quantity Purchased**, **Avg Spending per Transaction, Days Since Last Purchase**, **Customer Region**, and **Purchase Frequency** were used for clustering.
- **Number of Clusters:** 4 determine through Elbow Method:



  -
- **Preprocessing:** Data was normalized/standardized, One Hot Encoded, and any missing or outlier data was handled before applying the clustering algorithm.

3. Clustering Results

3.1 Cluster Hyperparameters:

- **Init:**
    - k-means++
- **max_iterations:**
    - 80
- **n_init:**
    - 3
- **n_clusters:**
    - 4

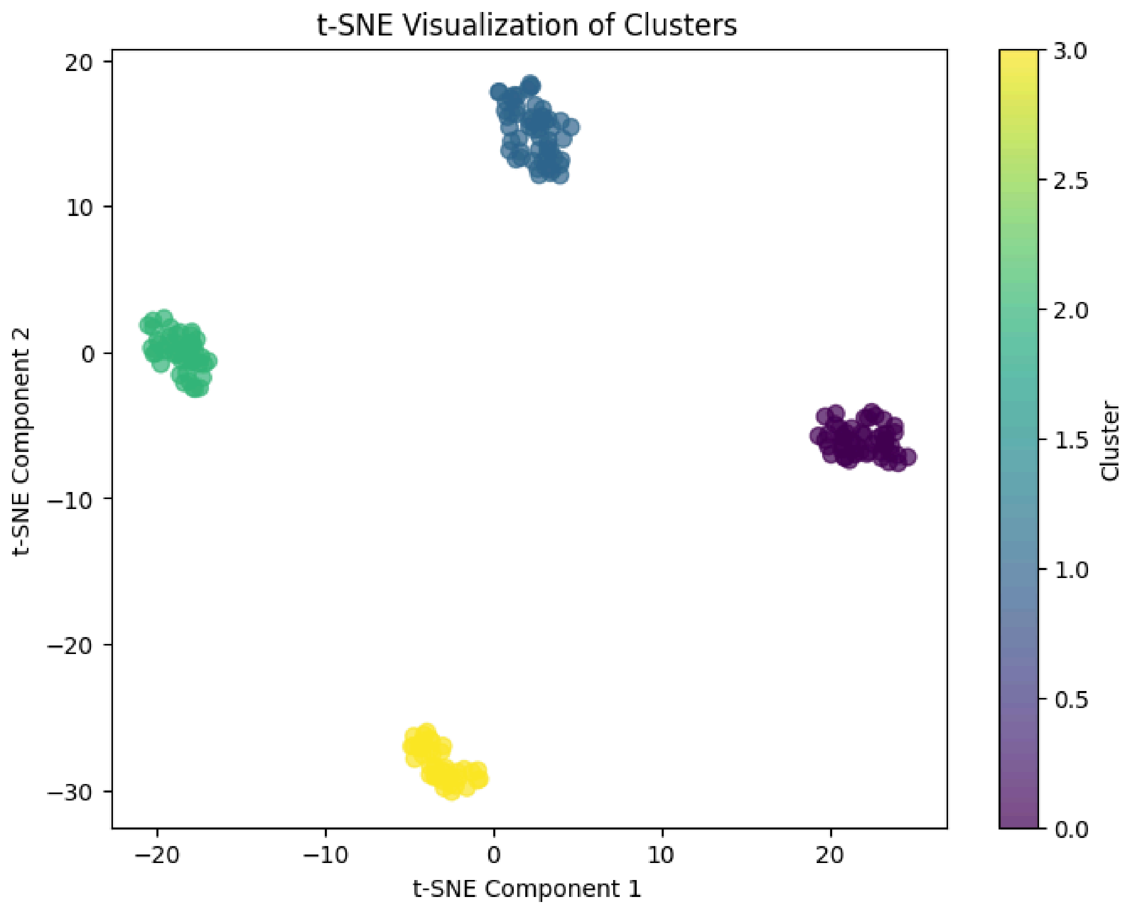
3.2 Clustering Metrics

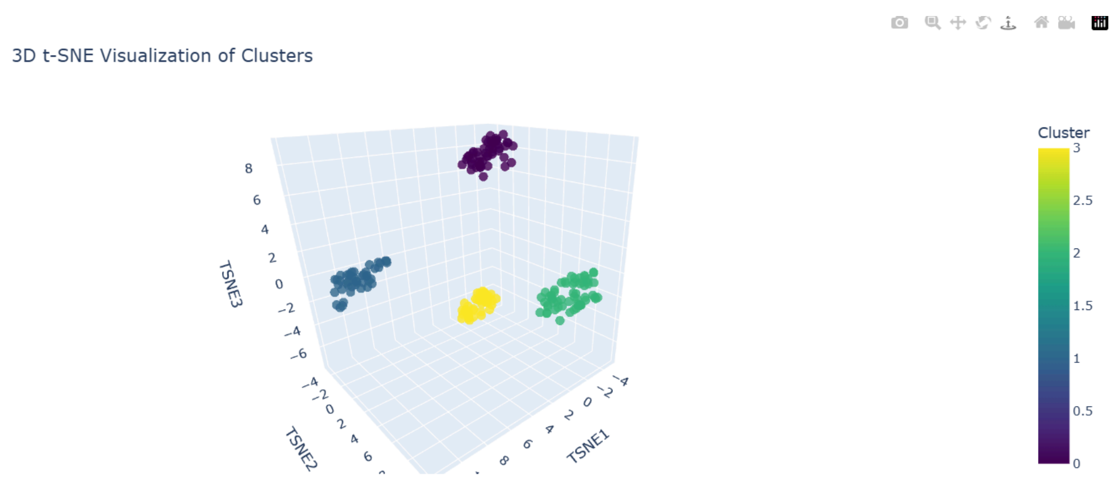
- **Davies-Bouldin Index: 0.4492**
- **Silhouette Score: 0.6983**

3.3 Insights from the Clusters

- **High-value customers** tend to buy more frequently and spend larger amounts. This group can be targeted with VIP-level promotions or early access to new products.
- **Bulk buyers** tend to make fewer, larger purchases but are more sensitive to discounts. Offering limited-time bulk discounts can help convert them into repeat buyers.
- **Low engagement customers** might require a more personalized approach, such as email campaigns or reactivation offers, to boost their purchasing behavior.

3.4 Clustering Plots:

- 



-

-Mohak Kapoor
+91 9870232056
contact.mohakapoor@gmail.com
[LinkedIn]()

GitHub